To the Graduate Council:

I am submitting herewith a thesis written by Harold H. Jones entitled "Comparison of Three Methods of Industrial Control Logic." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Robert E. Bodenheimer
Major Professor

We have read this thesis and recommend its acceptance:

Accepted for the Council:

Vice Chancellor
Graduate Studies and Research

# COMPARISON OF THREE METHODS OF INDUSTRIAL CONTROL LOGIC

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Harold H. Jones

June 1976

1286592

## ACKNOWLEDGMENTS

For his suggestion of the topic of this thesis the author is grateful to Dr. Robert E. Bodenheimer, Associate Professor of Electrical Engineering at the University of Tennessee.

## ABSTRACT

The purpose of this investigation was to obtain a comparison between the two types of industrial controls currently in use and to compare them with control using an Intel 8080 A microprocessor system. Cost, reliability and ease of modification are discussed for each system.

Comparisons indicate that the microprocessor, although new to the industrial control field, is now the outstanding method of control.

TABLE OF CONTENTS


| CHAPTER | | PAGE |
|---|---|---|
| I. | DEVELOPMENT OF THE PROBLEM AND TEST CASE | 1 |
| | Development of the Problem | 1 |
| | Development of Test Case | 3 |
| II. | DISCRETE LOGIC DESIGN | 7 |
| III. | PROGRAMMABLE CONTROLLER DESIGN | 12 |
| IV. | MICROPROCESSOR CONTROLLER DESIGN | 19 |
| | Development of Subroutines | 19 |
| | Development of Main Program | 21 |
| | Program Testing | 24 |
| V. | COMPARISON OF THE THREE SYSTEMS | 26 |
| | Cost Comparison | 26 |
| | Reliability Comparison | 30 |
| | Conclusions | 31 |
| BIBLIOGRAPHY | | 32 |
| APPENDIXES | | 34 |
| | Appendix A | 35 |
| | Appendix B | 41 |
| | Appendix C | 50 |
| VITA | | 75 |

LIST OF TABLES


| TABLE | | PAGE |
|---|---|---|
| 3.1. | Programmable Controller Test Instructions | 12 |
| 3.2. | Programmable Controller Set Instructions | 13 |
| 3.3. | Programmable Controller Jump Instructions | 15 |
| 3.4. | Programmable Controller Instruction Example | 17 |
| 4.1. | RAM Table Example | 20 |
| 4.2. | Microprocessor System Example | 22 |
| 5.1. | Cost of Small Programmable Controller System | 27 |
| 5.2. | Cost of Small Intel Microprocessor System | 27 |
| 5.3. | Cost of Small Discrete Logic System | 28 |
| 5.4. | Cost of Large Programmable Controller System | 28 |
| 5.5. | Cost of Large Intel Microprocessor System | 29 |
| 5.6. | Cost of Large Discrete Logic System | 29 |
| C.1. | The Microprocessor Program | 51 |

LIST OF FIGURES


| FIGURE | | PAGE |
|---|---|---|
| 1.1. | Configuration of Test Stand | 4 |
| 1.2. | Schematic for Test Stand | 6 |
| 2.1. | RTL NOR | 8 |
| 2.2. | AND Circuit and OR Circuit Implemented from NOR Gate | 10 |
| 2.3. | Circuit for Equation BLOW | 11 |
| 2.4. | Simplified Circuit for Equation BLOW | 11 |
| A.1. | NORPAK System for SP1 | 36 |
| C.1. | Schematic MCS-80 System Design Kit | 69 |
| C.2. | SDK-80 Memory Map | 74 |

# CHAPTER I

## DEVELOPMENT OF THE PROBLEM AND TEST CASE

### Development of the Problem

Decreased costs of microprocessors and logic components are rapidly changing the methods of industrial logic control. Most commonly used methods of logic control are as follows:

1. Relay logic
2. Discrete solid state logic gates
3. Programmable controllers

The latest logic control concept which is not widely used at this point is that of the microprocessor.

This thesis will compare three methods of logic control by implementing a simple system with discrete logic, a programmable controller, and a microprocessor. Since relay logic is not as reliable as the other methods, relay logic will not be included in this comparison. This comparison will include hardware costs, design costs, and costs associated with changes in design after system installation. The system to be implemented is a materials handling system having two sources of material to be conveyed to three destinations. The control will perform the following:

1. Set up routes between one or more sources to one destination.
2. Supervise the routes by checking to insure there is a clear path at all times.
3. Stop the system for loss of a path.

4. Force the operator to go through a fixed start-up sequence.

Comparisons relative to cost and performance are for the following three types of control:

1. NORPAK-discrete logic
2. Digital Equipment Company's I-1430 programmable controller
3. Intel SDK-80 microprocessor system

Industrial logic control systems are unique in that usually only one copy of each system is built prohibiting the justification of minimization techniques which eliminate 10 percent or more of the logic gates. In addition design time must be minimized because it cannot be spread over thousands of units as is the case with the design of commercial products. Industrial logic control systems must operate at start up time with a minimum amount of debugging time. From an economic viewpoint, a $5,000 control system should never delay the start-up of a $5,000,000 process.

After a process is running under the supervision of a logic control system, a full year may elapse before the logic controls can be modified or adjusted. The necessity for reliability and correctness in design forces the designer of industrial logic control systems to consider only proven logic elements and to use design techniques that have proven correct in the past.

Safety can be the largest factor in industrial logic controls as may be exemplified by combustion controls. In natural gas furnace systems, logic controls the gas valves, combustion air, ignition source, monitor flame, and supervises the purge time for a safe start. A mistake in

design or a failure of a logic component may result in disastrous consequences.

Development of Test Case

Test case for this thesis is an air conveying system with the following operating sequence (Figure 1.1 presents the configuration of the test system):

Step 1. Operator selects route to one of the three silos by pushing one of three momentary push button switches designated SRSL1, SRSL2, or SRSL3. Abbreviation stands for "set route to silo 1, 2, or 3." Assuming the operator makes a mistake at this point, a route cancel push button can be operated to immediately cancel the first selection. Only one route at a time is allowed after the operator has set a route. The logic control for each diverter valve decodes his request and energizes the correct diverter valves, and logic control then checks by means of limit switches to see if all diverter valves that affect a route are actually in the correct position.

When all limit switches define a correct route and agree with the route selected, a route set light will be energized on a graphic panel indicating to the operator that he has a correct route to the correct silo.

This process will energize a system route complete light which indicates to the operator that the blower start logic is enabled. The operator now has control of the blower by two pushbutton switches labeled START BLOWER and STOP BLOWER. These switches are abbreviated as STAR B and STOP B.

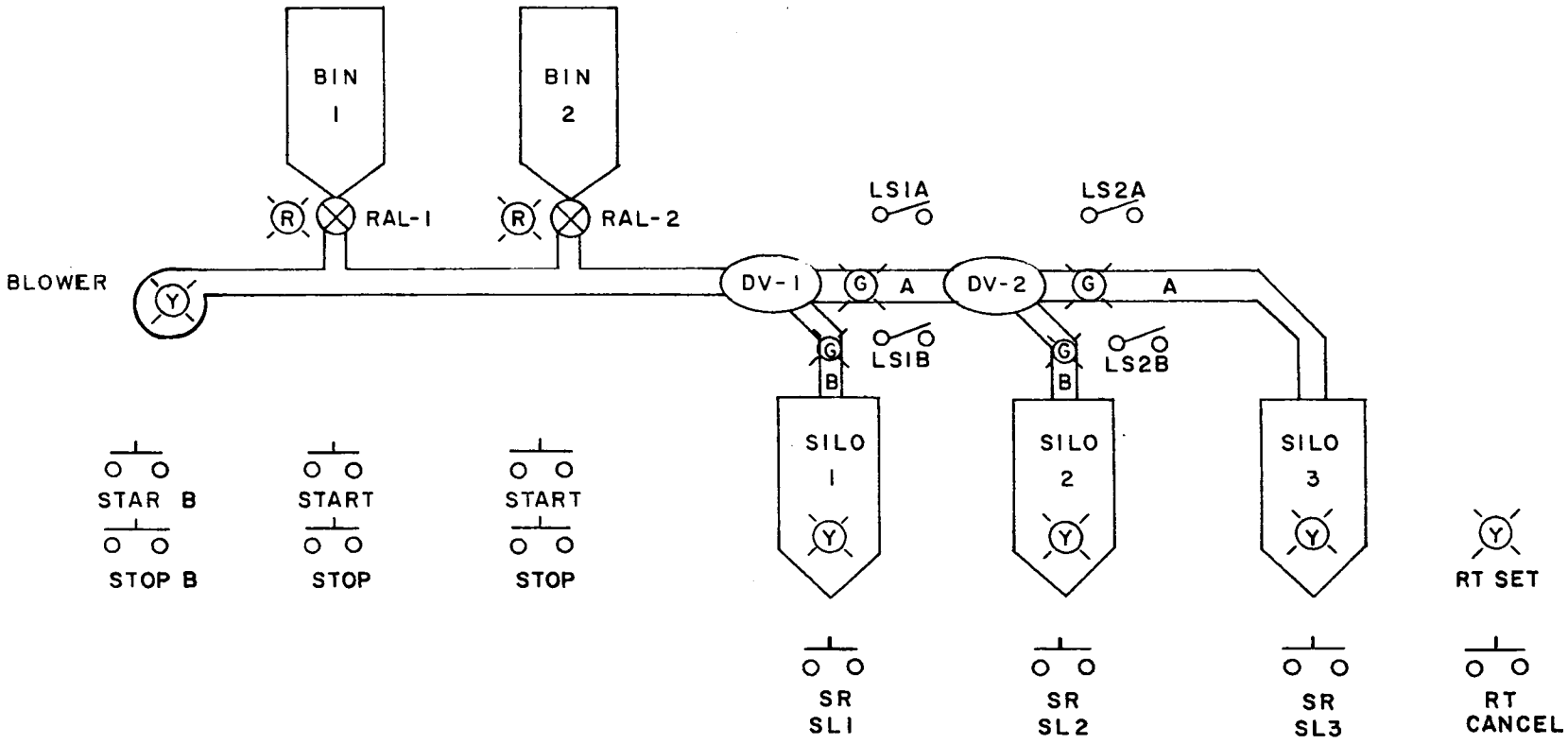



Figure 1.1. Configuration of Test Stand.

Step 2. Operator starts blower. After the blower is energized the rotary valve feeder logic is enabled.

Step 3. Operator starts one or both feeders to convey material to the selected source.

Step 4. Operator stops conveying by operating ROUTE CANCEL push button. When this happens, the logic control immediately shuts off all feeder valves and starts the flush timer labeled TD1. After a ten second time period, the blower is stopped and the present route cancelled. Note that the route is not cancelled immediately but remains set for the preset time period of the flush timer. In an emergency, the operator may stop the blower which will result in the shut down of the entire system; however, the route should remain set since an emergency stop indicates that conveying is not finished and material still needs to be flushed from the conveying system.

A test stand with lights for all outputs and switches for inputs was constructed to prove the logic of the three systems under investigation. Figure 1.2 presents the schematic for the test stand.

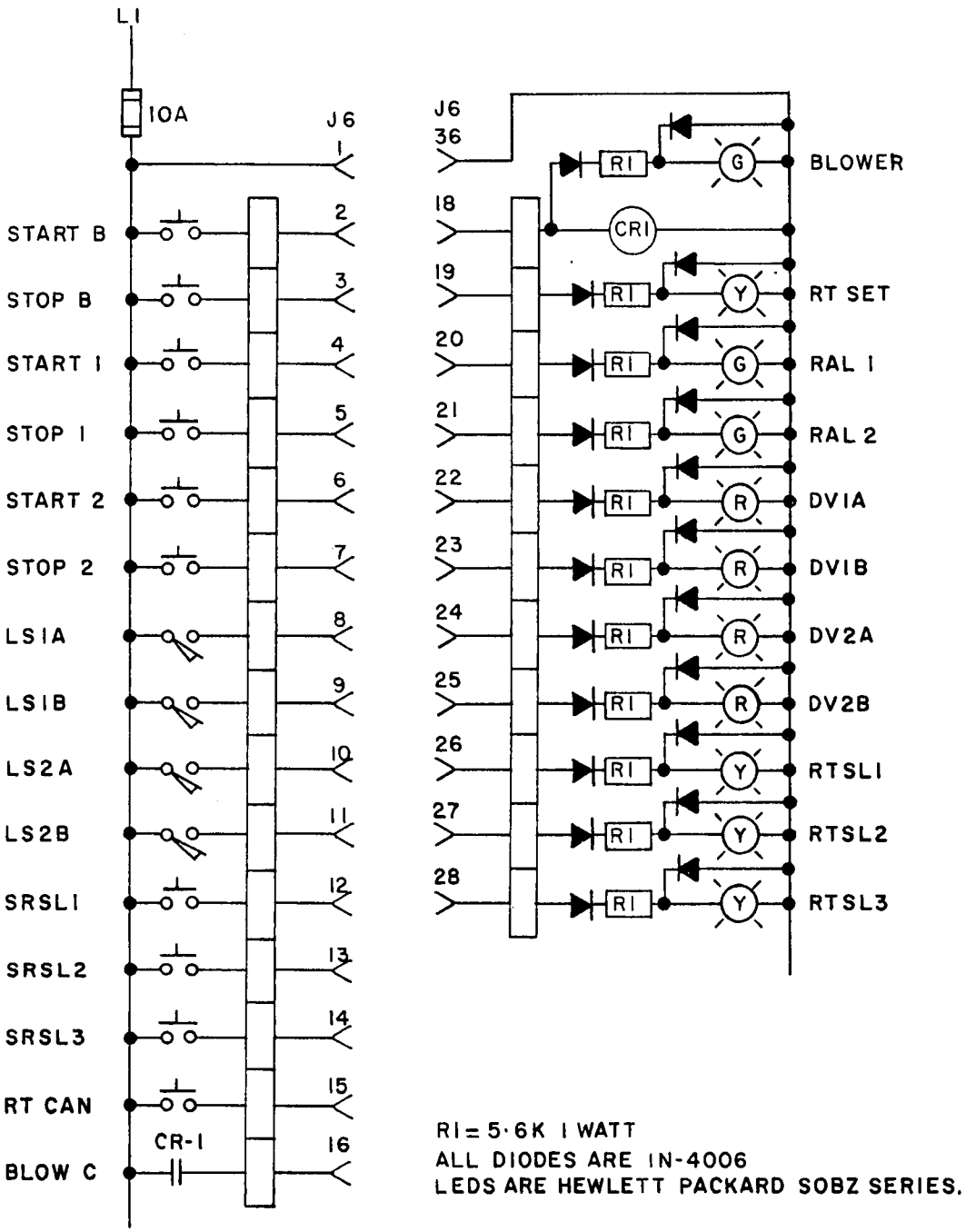



Figure 1.2. Schematic for Test Stand.

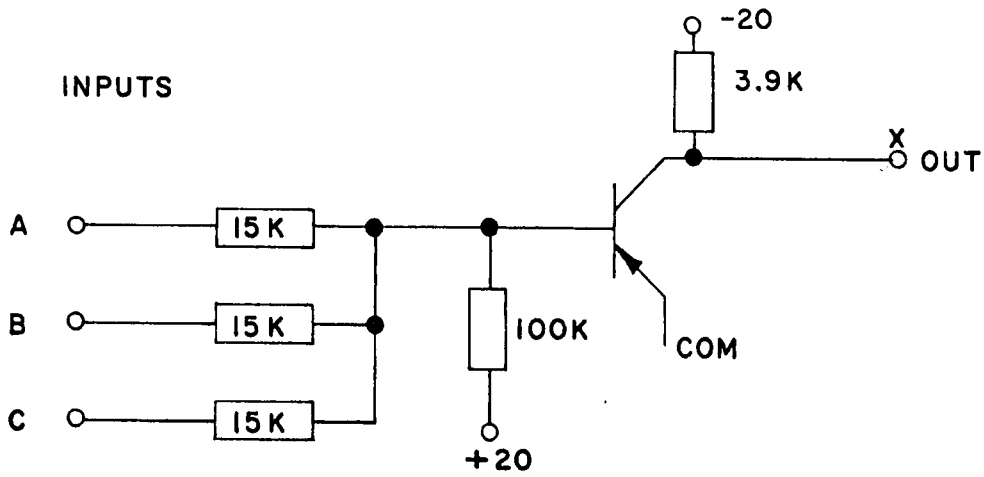
# CHAPTER II

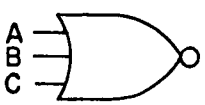
## DISCRETE LOGIC DESIGN

Discrete logic for control was implemented with the use of Square D Company's encapsulated solid state logic Norpak. Norpak line includes logical NOR'S, OR'S, timers, counters and various input-output components. Resistor Transistor Logic (RTL) is the basic type of logic used in Norpak. The primary advantage of this type of logic is its exceptionally high reliability. Disadvantages of Norpak include high costs incurred for preparing connection diagrams and costs associated with unauthorized field changes. In addition, some temperature problems occurred in earlier models due to the use of germanium transistors which are less heat resistant than newer silicon models. Use of silicon transistors in present Norpak systems has eliminated temperature problems. Figure 2.1 presents a circuit diagram for a basic NOR.

A major industrial control problem is the effect that noise has upon the system. Norpak has good noise immunity due to using a +20 volt supply to back bias the transistor to its off state. The Norpak design is straightforward and is implemented by interconnecting NOR and OR gates.

Elementary diagrams are first sketched by the designer and given to the drafting department where a complete connection diagram for the system will then be prepared. This connection diagram will be used by electricians in hand wiring all interconnections. See Appendix A for the elementary diagram of the test case.




NOR
LOGIC SYMBOL

TRUTH TABLE FOR 3 INPUT NOR

| C | B | A | X |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |

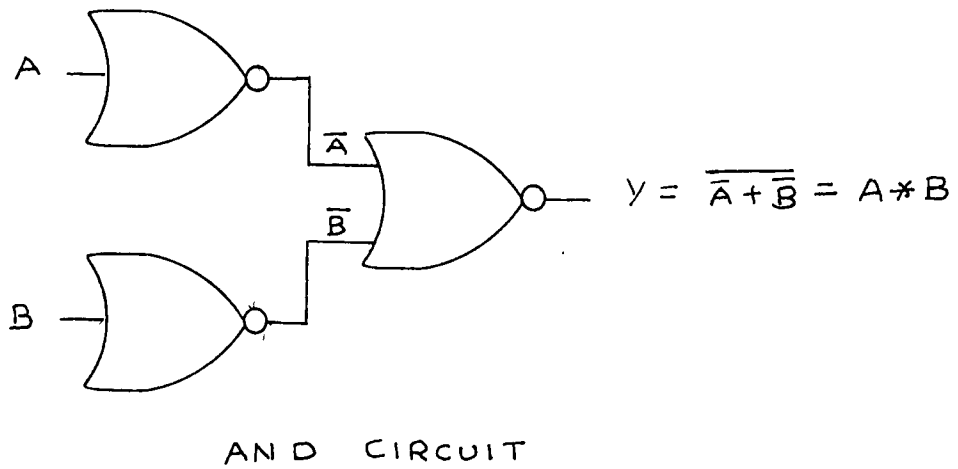
Figure 2.1. RTL NOR.

Design of discrete logic for industrial controls can be aided by first writing the Boolean logic equations for each output. Logic can be implemented directly by noting that LOGICAL AND, *, can be made from NOR'S by interconnection as presented in Figure 2.2.

LOGICAL OR, +, can be made by inverting the output of a NOR with an inverter or another NOR as presented in Figure 2.2.

Design of the Circuit for the output BLOW in the test case is presented in Figure 2.3.

This circuit can then be simplified through inspection as depicted by Figure 2.4. Minimization by inspection does not yield the minimum circuit in most cases, but for small specialized systems, inspection is the only method used. The Norpak technical manual contains several suggestions on minimization by inspection [8].



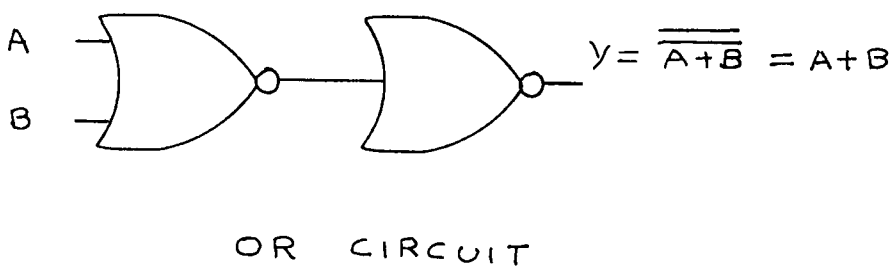
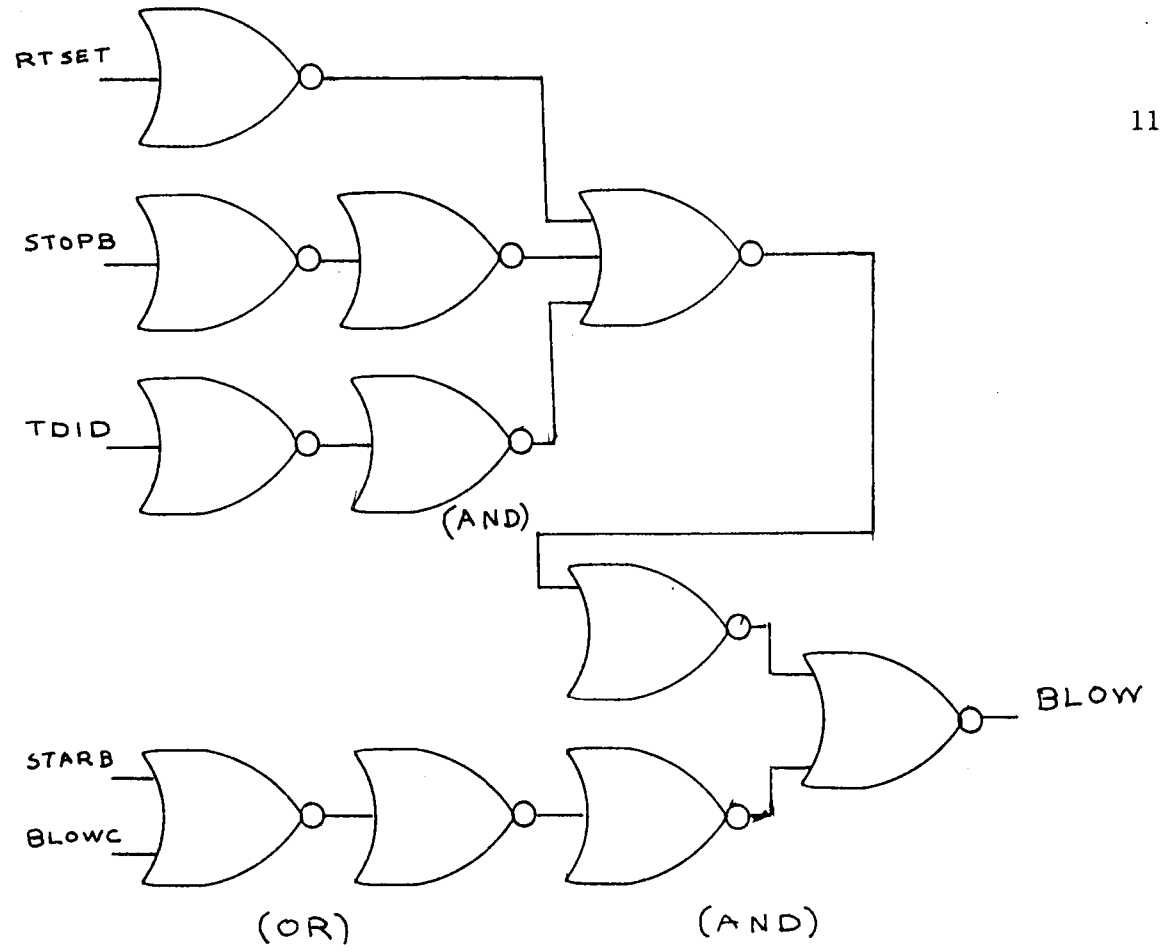




Figure 2.2 AND Circuit and OR Circuit implemented from NOR gate.




BLOW = RTSET * /STOPB * /TDID * (STARB + BLOWC)

Figure 2.3. Circuit for equation BLOW.

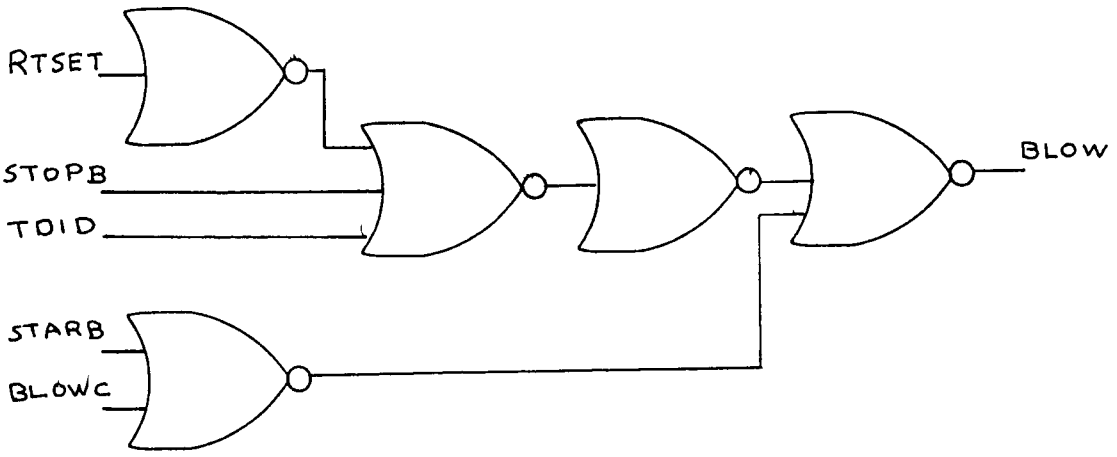



Figure 2.4. Simplified circuit for equation BLOW.

# CHAPTER III

## PROGRAMMABLE CONTROLLER DESIGN

The specific programmable controller used for purposes of comparison in this thesis is the I 1430 manufactured by Digital Equipment Company. Logic design for this controller is accomplished by writing Boolean expressions for each output. Equations may have subroutines for logic repeated in two or more equations, and may contain feedback elements to implement sequential logic circuits. There is no limitation on the length of equations, nor is the number of parens limited. The controller updates the status of all inputs at the beginning of the program, stores the status in temporary registers, and then solves each equation by testing elements in the equation for off or on status with a specialized instruction set [7]. Table 3.1 presents the instruction set for the I 1430.

Table 3.1

Programmable Controller Test Instructions

| SYMBOLIC | NUMERIC | DEFINITION |
|---|---|---|
| TF I/O | 4ØØØ + I/O | Test an I/O point for the OFF state. If the test is true and the test flag is currently OFF, the test flag is set ON otherwise the flag remains unchanged. |
| TN I/O | 6ØØØ + I/O | Test an I/O point for the ON state. If the test is true and the test flag is currently OFF, the test flag is set ON otherwise the flag remains unchanged. |

Logic equations for outputs are written using short symbolic labels of six or less characters for each element in the equation. The logical operators are:

* = AND

/ = NOT

\+ = OR

Industrial 14 basic instructions consist of three basic types:

1. Test Instructions—sample the current status of I/O registers
2. Jump Instructions—conditional or unconditional for executing instruction out of sequence
3. Set Instructions—set outputs ON or OFF.

Test instructions are used in conjunction with a test flip-flop or test flag. Symbolic and numeric forms and descriptions are given in Table 3.1 for the test instructions.

Set instructions turn an output ON or OFF at the end of a sequence of test instructions. Symbolic and numeric forms and descriptions are given in Table 3.2 for the set instructions.

TABLE 3.2

Programmable Controller Set Instructions

| Symbolic | Numeric | Definition |
|---|---|---|
| SN O | 2ØØØ + O | Set output O ON. O will remain on until set off by a SF O or CLR instruction |
| SF O | ØØØØ + O | Set output O OFF. O will remain OFF until set ON by an SN O instruction |

Jump instructions cause a departure from the normal sequential execution of instructions. Conditional jumps are used to jump to a SN or a SF instruction depending on the status of the test flag. Unconditional jumps are used to transfer control from one section of memory to another. Subroutine jump instructions are similar to unconditional jumps except the return address is saved and the subroutine is ended by a jump return instruction to return control to the main program.

The Industrial 14 allows only one level of subroutines because only one return address may be saved at a time. Symbols, numerals and definition for Jump instructions are presented in Table 3.3.

Two instructions that do not fall under Test Set or Jump Instructions are MOVEBIT and NOP. MOVBIT is an instruction used only in shift register circuits and moves I/O status from one location to another. NOP is an instruction that causes no action and is used in field editing to remove elements of an equation or is replaced by other instructions to add elements to an equation.

Appendix B contains the complete program for the test case. Equation One will be presented here for analysis.

Symbolic notation for Equation One:

BLOW = RT SET * /STOPB * /TDID * (STARB + BLOWC)

The "BLOW" output is on when RTSET is on and the STOP PUSH BUTTON is not depressed, TDID is not timed out, the start push button is depressed, or the blower is already running.

The segment of the equation (STARB + BLOWC) makes the equation sequential because the past status must be known to describe its present

TABLE 3.3

Programmable Controller Jump Instructions

| Symbolic | Numeric | Definition |
|---|---|---|
| JMP<br>NNNN | ØØ24<br>NNNN | Unconditional Jump to location NNNN. Takes two memory locations. |
| SKP | 010 | Skip the next memory location unconditionally. |
| CLRPC | 0004 | Unconditional Jump to address ØØØØ used at the end of program to reset the program counter to Ø. |
| JMS<br>NNNN | 0124<br>NNNN | Unconditional Jump to location NNNN. Takes two memory locations. Used to Jump to a subroutine. |
| JMR | 0054 | Unconditional Jump to two locations after the JMS instruction. JMR and JMS instruction must be paired. |
| JFF NNN | 2ØØØ + NNN | Jump to Location NNN if the test flag is OFF. Continue sequentially if the test flag is ON. The test flag is set OFF regardless of its original state. |
| JFN-NNN | 2400 + NNN | Jump to Location NNN if the test flag is ON. Continue sequentially if the test flag is OFF. The test flag is set OFF regardless of its original state. |

state. Assembler program BOOL I 143/8 requires that I/0 numbers or numeric notation be used in palce of symbolic notation. Conversion Program SET I 143/8 converts symbolic notation to numeric notation while retaining the symbolic notation as a comment.

Numeric notation for Equation One:

1000 = 1001 * /1 * / 16Ø1 * (Ø + 16)

Table 3.4 is an example of how the I 1430 specialized instruction set solves the Booleon expression for "BLOW" under three different situations. Case 1 is for all I/O off at the start of the program. Since RTSET or, 1001, is off and is a requirement before the equation can be true, the test flag is set on and at location 4 a jump if on instruction is executed to location 12. If output "BLOW" is on location 12 contains a SET OFF instruction to turn off the output. In Case 2 of Table 3.4 "SETRT," 16Ø1, and "STARB," Ø, are on with all the I/O off at the start of program execution. In Case 2 all requirements are met for a true output for the equation "BLOW." In Case 2 the test flag remains off as the program executes sequentially to location 5. At location 5 a jump if off to location 10 instruction is executed. At location 10 a set on instruction is performed to turn on BLOW. Location 11 contains a skip the next location instruction which causes a jump to Location 13. Case 3 is for the situation where BLOW, 1ØØØ, is on and STARB, Ø, if off and BLOWC, 16 is on. Case 3 illustrates the resetting of the test flag when a jump instruction is encountered at Location 5 but not executed. The program moves to location 10 where the set on instruction is executed.

Table 3.4

Programmable Controller Instruction Example

| Location | INS. | I/O | CASE 1 Test Flag 1ØØØ | | CASE 2 Test Flag 1ØØØ | | CASE 3 Test Flag 1ØØØ | |
|---|---|---|---|---|---|---|---|---|
| Ø | TF | 1ØØ1 | 1 | Ø | Ø | Ø | Ø | 1 |
| 1 | TN | 1 | 1 | Ø | Ø | Ø | Ø | 1 |
| 2 | TN | 16Ø1 | 1 | Ø | Ø | Ø | Ø | 1 |
| 3 | JFN | 12 | 1 | Ø | Ø | Ø | Ø | 1 |
| 4 | TF | Ø | | | Ø | Ø | 1 | 1 |
| 5 | JFF | 1Ø | | | Ø | Ø | Ø | 1 |
| 6 | TF | 16 | | | | | Ø | 1 |
| 7 | JFN | 12 | | | | | Ø | 1 |
| 1Ø | SN | 1ØØØ | | | Ø | 1 | Ø | 1 |
| 11 | SKP | | | | Ø | 1 | | |
| 12 | SF | 1ØØØ | Ø | Ø | | | | |
| 13 | NOP | Ø | Ø | Ø | Ø | 1 | Ø | 1 |

Since "BLOW," 1000 was already on, it remains on. The skip instruction is executed as in Case 2.

Case One assume: all inputs and outputs OFF at the start of program execution.

Case Two assume: 1ØØ1 and Ø are ON at the start of program execution.

Case Three assume: 1ØØ1 remained ON and input 16 is now on. All other I/O in equation remain off.

The complete program for the test case is presented in the Appendix C.

A symbol table is used to assign each label an input or output number which is used by SET 143/8 to convert equations written with labels into Input/Output numbers [7]. The result of SET 143/8 is a listing of equations in both symbolic labels and I/O numbers. This listing is used as an input to Digital Equipment Company Industrial 14 Program BOOL 143/8 which writes the assembler level I 1430 language to solve the Boolean equations. BOOL 143/8 gives a listing of results along with error messages on any incorrectly formated equations. One pass through BOOL 143/8 will locate most errors in formating. The original source program is then edited with the 8E Test editor program and rerun through SET 143/8 and BOOL 143/8. An error free program is then ready to be debugged by the use of the ODP 143/8 program running on the 8E computer. Simulation of the program can be as complete as the designer desires, or as time constraints will allow. Inputs are simulated from the teletype keyboard, and outputs checked for correct switching.

# CHAPTER IV

## MICROPROCESSOR CONTROLLER DESIGN

The microprocessor program for solving Boolean equations was written using a Digital 8E computer Text Editor [9] and an INTEL Intellect 8 assembler [3]. An Intel 8080 A microprocessor was used as the central processing unit.

### Development of Subroutines

Purpose of the first part of the program is to store in memory the status of all inputs and outputs. Following storage, the program will recall from memory specific input and output status to solve a Boolean algebra expression for each output. A memory map of the SDK-80 system is presented in Appendix C. The stack, which is a storage block for temporary information, is not used extensively in this program and requires only sixteen locations. Timers require three RAM locations each while one location, 1300H, is used as temporary storage for building an output byte.

The Symbol Table defines characters and assigns a Symbol Label of one to five characters for each input, output, or timer and assigns each a location in Random Access Memory (RAM). Eight locations in memory are assigned each I/O port in consecutive order with no unused locations. Therefore, all RAM locations are assigned when the locations for the first input and output are assigned in the Symbol Table. The SDK-80 system has a total of 1024 RAM locations for status retention.

The first subroutine "INTR" is an interrupt subroutine called by the hardware for the purpose of updating timers. After storing the status of the 8080A CPU registers and flags, the INTR subroutine calls all timer subroutines. The hardware divides the clock frequency of 18,432 MHZ by 1,843 MHZ to give an interrupt signal at a 10 HZ rate or every 0.1 second. After the last timer has been called, the INTR subroutine restores the status of all registers and flags and returns to the main program with the interrupt flip-flop enabled.

Each timer subroutine checks the timer on-off status. When a timer is off, the subroutine loads full preset times into the RAM Preset Table, and then returns to INTR subroutine.

The IPTOM and MOVE subroutines move the status of an input port to I/O RAM Table one bit at a time. For example, Port A Chip 0 = 10001101 will be moved to the I/O RAM Table as shown in Table 4.1.

Table 4.1

RAM Table Example

| Ram Location | |
|---|---|
| 1320 - 1000 0000 | PORTA CHIP 0 = 1000 1101 |
| 1321 = 0000 0000 | |
| 1322 = 1000 0000 | |
| 1323 = 1000 0000 | |
| 1324 = 0000 0000 | |
| 1326 = 0000 0000 | |
| 1327 = 1000 0000 | |

The 0Ø through 07 subroutines move as the result of a Boolean equation to the I/O RAM Table and build one bit of an output byte for each Boolean equation solved (see Table 4.2 for an example). Note that the output byte must be complemented because the output amplifiers are inverters.

Development of Main Program

At the beginning of the program, the stack pointer is set close to the top of the available RAM location. Routine INIT initializes 0000 0000 in all used I/O RAM locations at power-up, and all programmable ports are set to input ports. Routine INIT then sends a control word to all programmable output ports to set them to output mode. LOOP is the location of the normal start of the program after the first pass through the program. The last instruction in the program is a jump to LOOP. At LOOP, the location of the first input is loaded in H and L registers.

The first port status is called with the "IN" instruction to the accumulator or A register. Subroutines "IPTOM" and "MOVE" then move the status of the first eight input points to the I/O RAM Table. Status of all remaining input ports is similarily moved to the I/O RAM Table. At this point in time, the RAM Table has the current status of all inputs, outputs, and timers.

The Boolean equations are solved by loading the off-on status of each symbolic label in the equation into the registers and then performing the indicated logical AND, OR, EXCLUSIVE OR, and NOT as indicated by the equation. A complete set of instructions for the 8080A microprocessor can be found in the INTEL 8080 microcomputer systems user's manual [4].

Table 4.2

Microprocessor System Example

| Instruction | | Comment |
|---|---|---|
| LDA RTSET | ; | Load the status from RAM Table into the accumulator register |
| MOV B, A | ; | Move the status of RTSET from the accumulator to register B |
| LDA STOPB | ; | Load the status of STOPB from RAM Table into the accumulator register |
| CMA | ; | Complement the status of STOPB to give /STOPB status |
| MOVE B, A | ; | Move the status of /STARB to register B |
| LDA TDID | ; | Load the status of TDID from RAM Table into the accumulator register |
| CMA | ; | Complement the status of TDID to give /TDID |
| ANA B | ; | Logical and /TDID in Register A with RTSET in Register B and put the result in register A |
| ANA C | ; | Logical and /TDID * /RTSET with /STOP B in Register C and put the result of /TDID * /RTSET * /STOP B in Register A |
| MOV D, A | ; | Move the result of RTSET * /STOP B * /TDID to Register D for temporary storage |
| LDA STARB | ; | Load the status of STARB from RAM table into the accumulator register |
| MOV B, A | ; | Move the status of STARB to Register B |
| LDA BLOWC | ; | Load the status of BLOWC from RAM Table to the accumulator register |
| MOV B, A | ; | Move the status of BLOWC to register B |
| OR A B | ; | Logical OR STARB with BLOWC and put the result in the accumulator or register A |

Table 4.2 (continued)

| Instruction | | Comment | | |
|---|---|---|---|---|
| ANA D | ; | Logical and STARB + BLOWC with RTSET * /STOPB * /TDID and put the result in the accumulator | | |
| Call 0Ø | ; | Call subroutine 0Ø which stores the status of BLOW until eight equations have been solved | | |
| 000 | | 0000 | 000X | X represents the status of BLOW in the Byte to OUTPUT PORT CHIP 1 Port A |
| Call 01 | | 0000 | 00X0 | |
| Call 02 | | 0000 | 0X00 | |
| 0 | | | | |
| 0 | | | | |
| 0 | | | | |
| Call 07 | | X000 | 0000 | |
| OUT 10EC | ; | Sends the result of 8 equations to an output port Chip 1 Port A where the status is latched until the next sequential pass through the program | | |

Program Testing

After the program has been written with the text editor and assembled by the resident MACRO ASSEMBLER [3], the program is ready to be debugged. The debugging operation is accomplished through the following steps:

1. Load program into RAM in the Intellect 8.
2. Change the jump to LOOP to jump to SIM at the end of program.
3. Simulate by changing the status of the inputs in RAM Table and observing the outputs. The S for substitute memory command in the Resident Monitor is used primarily for this purpose.

SDK-80 kit contains a system monitor program for debugging programs in RAM. Monitor commands include Display memory, Substitute memory, Insert into memory, Move memory, and Goto memory location commands. The SDK-80 has provision for 1K bytes of RAM which is sufficient for most subroutines and small programs. Program debugging is accomplished by loading individual subroutines into RAM and testing portions of the program by use of the Restart command RST 1. RST 1 is substituted for the normal command at a point under investigation. When RST-1 is reached in the sequential execution of commands, control is returned to the monitor program with all flags and register status saved. The Monitor Examine command will then dispaly the status of all CPU registers and flag conditions. Problems can easily be traced with the RST-1 feature [6]. After all subroutines and the main program are checked, they can be combined into a programmable read only memory (PROM). SDK-80 monitor does not have the capability to program a PROM. Programming of PROM can be done on a longer computer

such as an Intel Intellect 8 or Digital Equipment Company's PDP-8 with a PROM programming board. SDK-80 has provision for 4K bytes of PROM without memory extension boards, but memory can be extended to the 65K bytes capability of the CPU. For industrial logic controls, 4K of PROM and 1K of RAM is almost always adequate. Schematic of the microprocessor system and fail safe shutdown circuit is presented in Appendix D.

# CHAPTER V

## COMPARISON OF THE THREE SYSTEMS

### Cost Comparison

Cost comparison for the small system test case from Chapter II has been developed by fully implementing the design and keeping records of time used. Tables 5.1 through 5.3 present cost comparison for the three systems.

The Norpak system was fully designed but was not actually built. Labor costs required for developing the Norpak system have been calculated based on similar jobs with similar designs.

Microprocessor design includes labor costs to build the system and add the necessary hardware for timer interrupt and power fail restart circuits. Development costs for microprocessor subroutines are not included since the subroutine is general in nature and could be used for other systems thus spreading initial costs over a large number of systems. Input/Output costs as well as costs for interconnecting cables are also included.

The programmable controller design was fully built and tested. Since the first design worked as planned, minimum labor costs were sustained. Large system costs have been estimated by scaling labor costs and including additional required hardware costs. Tables 5.4 through 5.6 present costs for the three designs.

Based on cost comparison, the microprocessor design is the most economical for both the large and small system.

Table 5.1

Cost of Small Programmable Controller System

| DEC | | | | |
|---|---|---|---|---|
| I1430 | Basic CPU | | $ 3,600.00 | |
| | (16) Input Converts 29.69 ea. | | 475.04 | |
| | (16) Output Amplifers 35.91 ea. | | 575.04 | |
| | | | $ 4,650.08 | |
| | | Labor | 78.75 | |
| | | | $ 4,728.83 | Total |
| | Labor $15/hr. | Designing | One hour | |
| | | Drafting | Four Hours | |
| | | Checkout | Twenty-five Minutes | |

Table 5.2

Cost of Small Intel Microprocessor System

| INTEL<br>SDK-80 | | | | |
|---|---|---|---|---|
| SDK-80 KIT | | | $ 350.00 | |
| Additional logic components | | | 20.00 | |
| Power Supply | | | 97.50 | |
| Box, Cables | | | 155.75 | |
| (16) Output Amplifers 17.39 ea. | | | 203.52 | |
| (16) Input Converters 12.72 ea. | | | 118.24 | |
| | | | $ 945.01 | |
| | | Designing | 360.00 | |
| | | Four Hours | $ 1,305.01 | Total |
| Does not include | | Drafting | | |
| original subroutines | | Four Hours | | |
| and format for the | | Checkout | | |
| first program | | Four Hours | | |

Table 5.3

Cost of Small Discrete Logic System

| SQUARE D NORPAK | | | |
|---|---|---|---|
| (2)-20 Pack | 90.10 each | $180.20 | |
| (1)-OR | 31.80 | 31.80 | |
| (16) Output Amp | 21.20 | 339.20 | |
| (16) Input Converter | 12.72 | 203.52 | |
| (1) TD | 63.60 | 63.60 | |
| (1) Power Supply | 116.60 | 116.60 | |
| | | $934.84 | |
| | 40 hr. construction time | | |
| | Design time One Hour | 600.00 | |
| | 40 dr. drafting | 630.00 | |
| | Check-out Time One hr. | | |
| | | $2264.84 | Total |

Table 5.4

Cost of Large Programmable Controller System

| DEC I1430 | | | |
|---|---|---|---|
| | CPU | $ 3,600.00 | |
| Input Converters | 256 x 29.69 | 7,600.64 | |
| Output Amplifiers | 256 x 35.94 | 9,200.64 | |
| | | $20,401.28 | |
| Labor $16/hr. x small system | | 1,260.00 | |
| | | $21,661.28 | Total |

Table 5.5

Cost of Large Intel Microprocessor System

| INTEL SDK-80 | | | |
|---|---|---|---|
| | SDK 80 Kit | 350.00 | |
| | Additional logic components | 230.00 | |
| | 1K RAM 6K Men PROM | 330.00 | |
| | Connectors & Cable | 350.00 | |
| | Output Amplifiers 256 x 17.39 | 1,891.84 | |
| | Input Converters 256 x 12.72 | 3,256.32 | |
| | | $ 6,588.11 | |
| | Labor $16/hr. x small system | 5,760.00 | |
| | | $12,348.11 | Total |

Table 5.6

Cost of Large Discrete Logic System

| SQUARE D NORPAK | | | | |
|---|---|---|---|---|
| (32) | 20pak | 90.10 ea. | $ 2,883.20 | |
| (16) | OR | 31.80 ea. | 508.80 | |
| (256) | Output Amplifiers | 21.20 ea. | 5,427.20 | |
| (246) | Input Converters | 12.72 ea. | 3,256.32 | |
| (16) | Time Delay | 63.60 ea. | 1,017.60 | |
| (2) | Power Supply | 16.60 ea. | 233.20 | |
| | Construction | 16 x 40 x 15 | 9,000.00 | |
| | | | $22,926.32 | |
| | | Labor $16/hr. x small_system | 10,080.00 | |
| | | | $33,006.32 | Total |

Reliability Comparison

The reliability of the programmable controller has varied with different units with central control units experiencing some chip failure. Most problems have been associated with I/O failure. The programmable controller itself contains approximately 500 integrated circuits.

> A typical 16 pin I C will introduce approximately 36 interconnections in a system (16 interconnections from the chop to the lead frame, 16 from the lead frame to the p.c. card, two interconnections from the p.c. card to the back plane, and two interconnections from back-plane point to back-plant point). A digital system fails most frequently because interconnections fail [2].

A potential problem exists with the approximate eighteen thousand interconnections of the present generation programmable controllers.

The reliability of microprocessors in industry is not completely known, but the number of interconnections is much less than that of the programmable controller. The total number of interconnections on the microprocessor design is approximately one-thousand compared to 18,000 for the programmable controller. As a result of the smaller number of interconnections, the microprocessor should be significantly more reliable than the programmable controller.

Design changes after an industrial logic system is implemented and the original designer is not available makes small modifications of the system difficult and time consuming. In some instances of Norpak design, the logic is completely redesigned for a small change rather than spending a large amount of time deciphering unfamiliar logic diagrams.

The programmable controller simplifies the problem of following another designer primarily because the logic is stated in individual

Boolean algebra equations for each output. This method of design breaks the logic system down into small understandable parts.

## Conclusions

When the total system cost is compared, the microprocessor design is the best choice regardless of system size. Reliability is a must in industry for safety as well as maintenance cost. From a reliability viewpoint, the microprocessor design is better than the programmable controller design, but no better than the Norpak design. Security of the system from unauthorized change is excellent for the microprocessor and the programmable controller and poor for the Norpak design. Ease of design change would be best with the programmable controller and worst with Norpak.

When the system is considered from a total viewpoint of cost, reliability, security from unauthorized change and ease of design changes, the system implemented with the microprocessor is the best approach.

# BIBLIOGRAPHY

## BIBLIOGRAPHY


1. COS/MOS DIGITAL Integrated Circuits, RCA Corporation, Somerville, N.J., 1975.

2. Davidson, William. Microprocessor Round-Up. New York: McGraw-Hill Book Company, Inc., 1975.

3. INTEL 8080 Assembly Language Programming Manual. INTEL Corporation, 3065 Bowers Avenue, Santa Clara, Calif.

4. INTEL 8080 MICROCOMPUTER Systems User's Manual, SRP, 1975. INTEL Corporation, 3065 Bowers Avenue, Santa Clara, Calif.

5. INTELLECT 8 System User's Manual. INTEL Corporation, 3065 Bowers Avenue, Santa Clara, Calif.

6. INTEL MSC-80 System Design Kit User's Guide. INTEL Corporation, 3065 Bowers Avenue, Santa Clara, Calif.

7. INDUSTRIAL 14 Software Manual. Digital Equipment Corporation, Maynard, Mass.

8. Norpak Technical Manual. Square D. Company, Milwaukee, Wis.

9. PDP 8 System User's Handbook. Digital Equipment Corporation, Maynard, Mass.

10. Wood, Paul E. Switching Theory. New York: McGraw-Hill Book Company, Inc.,

# APPENDIXES

# APPENDIX A

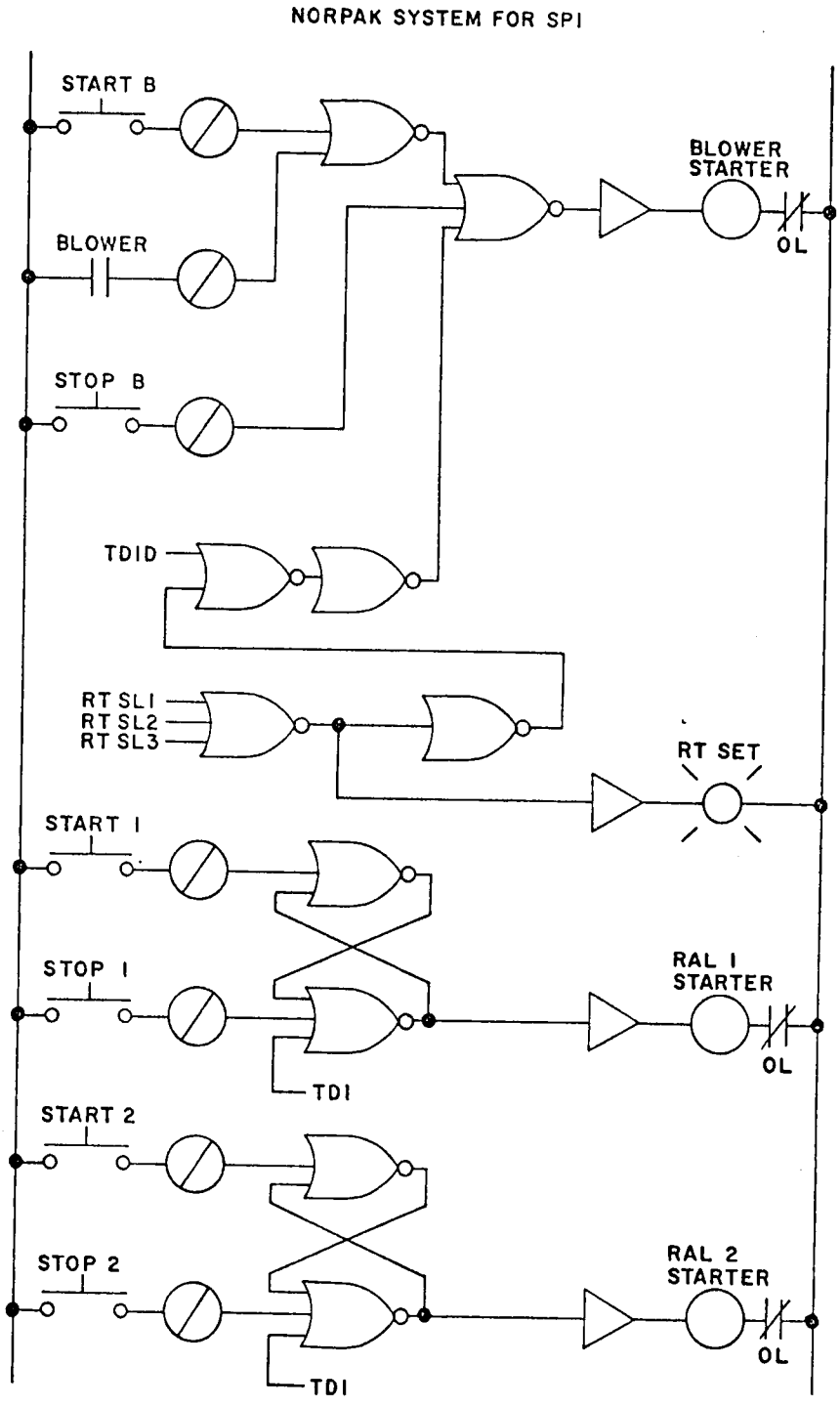



Figure A.1. NORPAK System for SP1.

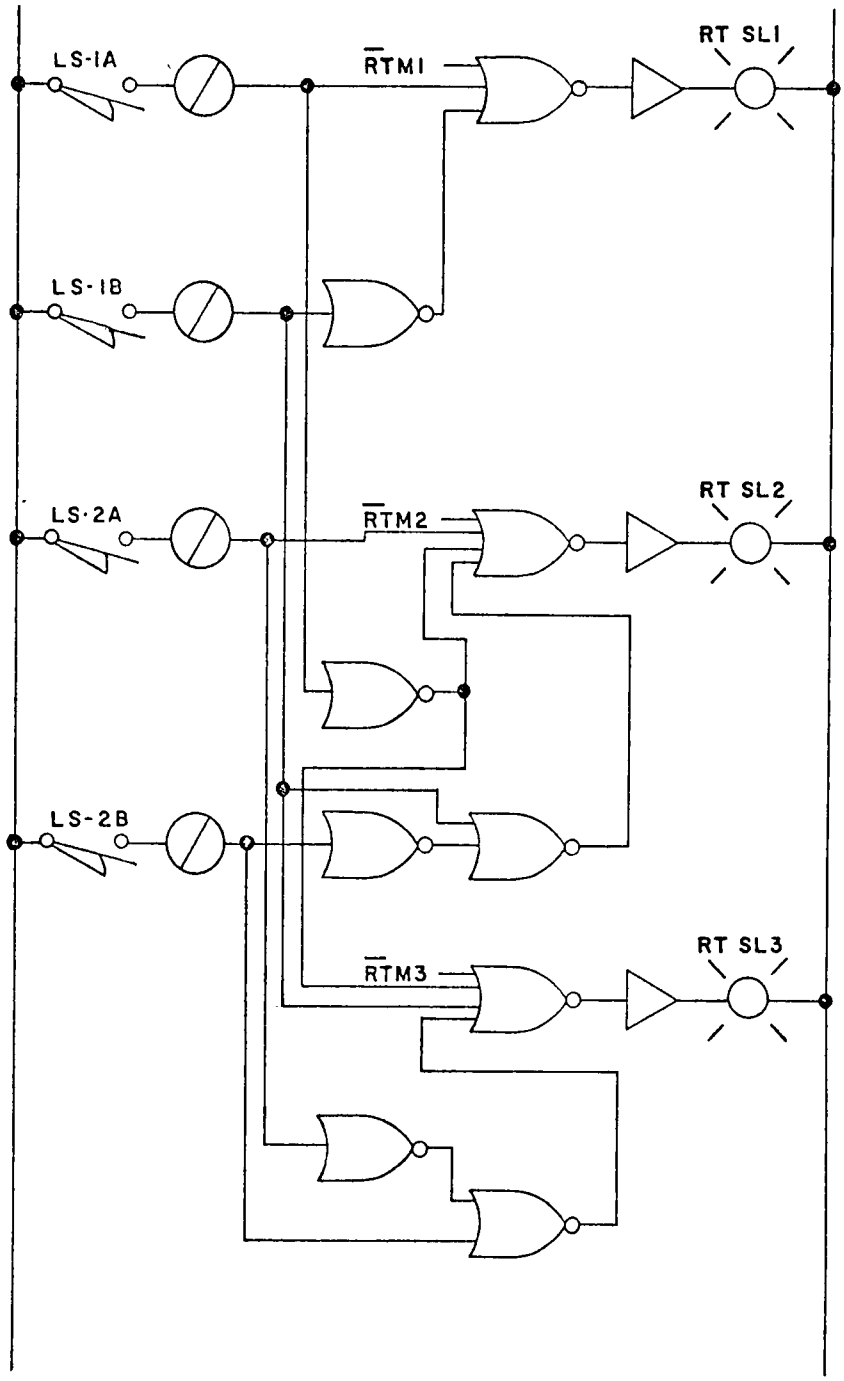



Figure A.1 (continued)

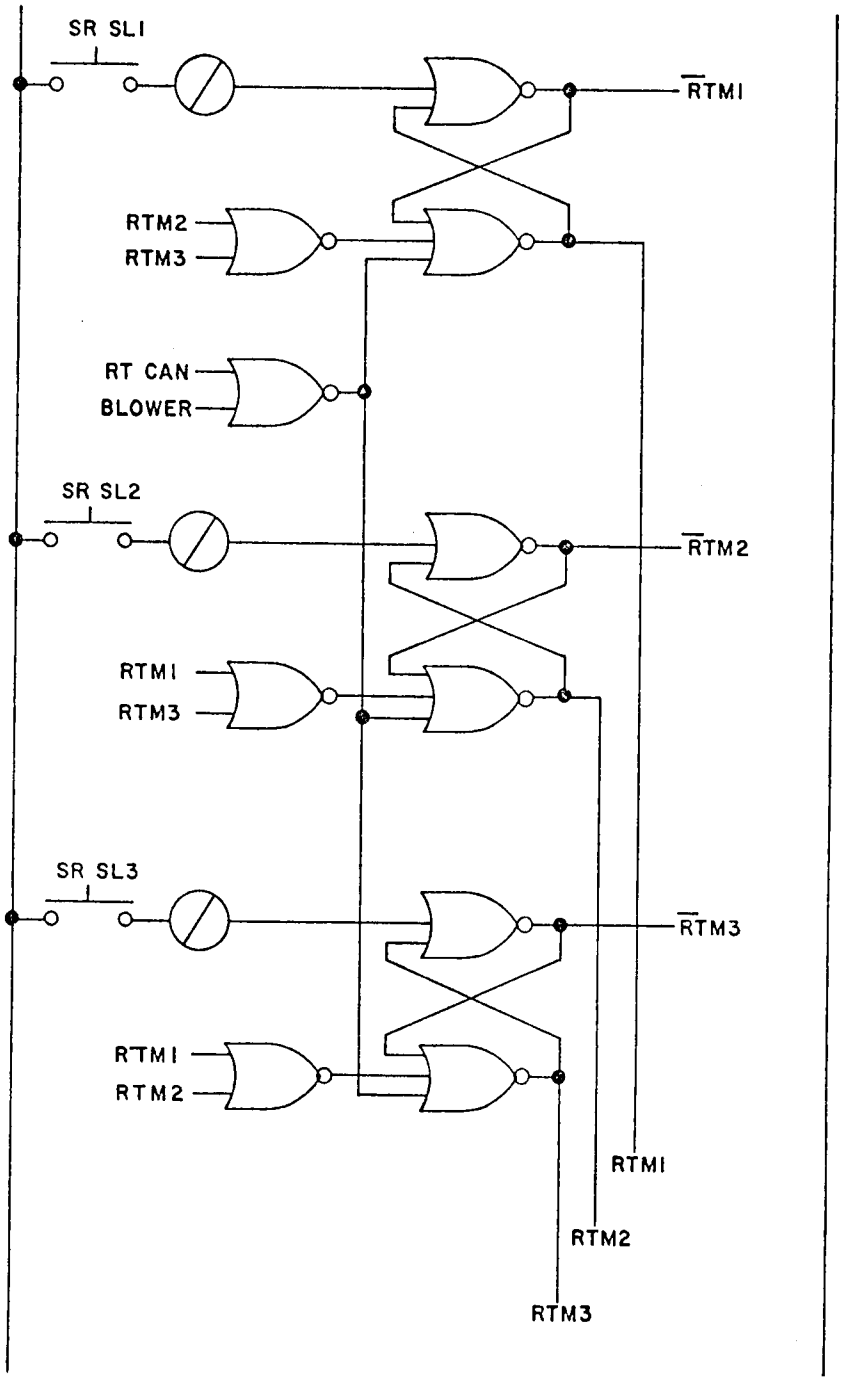



Figure A.1 (continued)

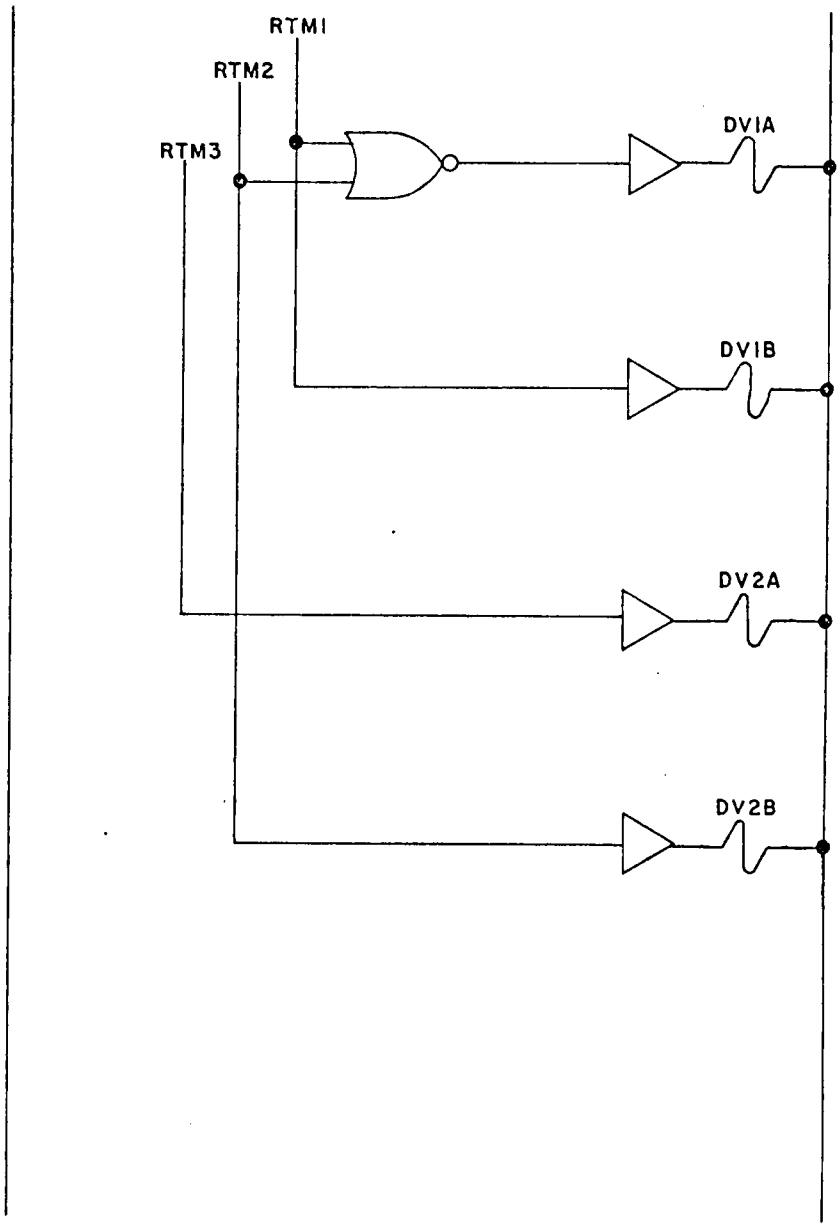



Figure A.1 (continued)

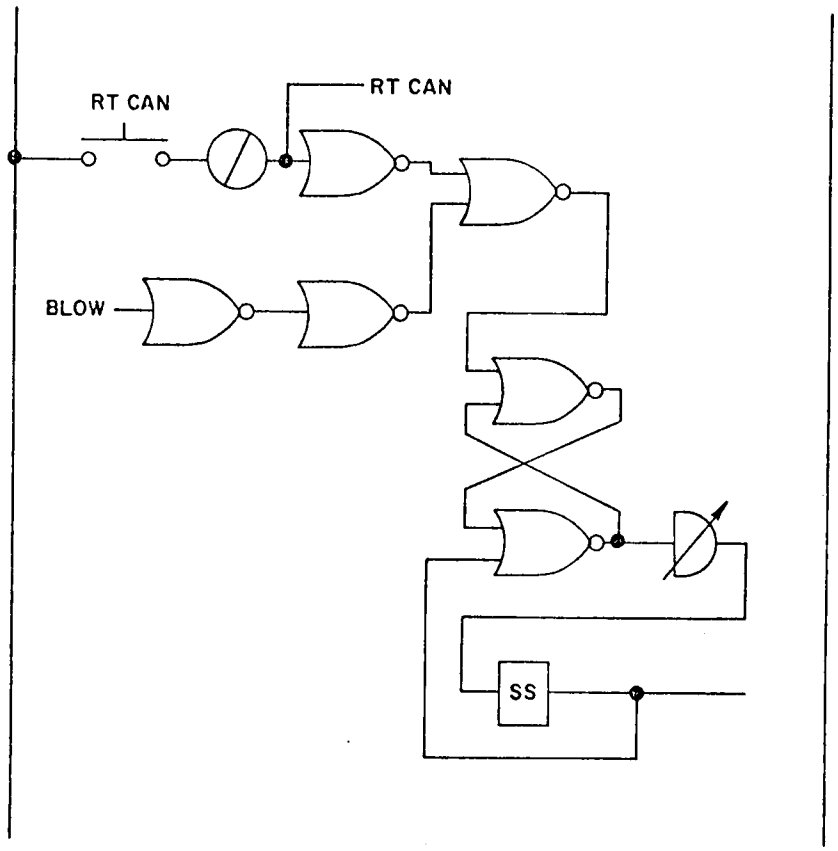



Figure A.1 (continued)

# APPENDIX B

```
;SYMBOL TABLE FOR EACH LOGIC ELEMENT


;$
#
;$
STARB=   0
;$
STOPB=   1
;$
STAR1=   2
;$
STOP1=   3
;$
STAR2=   4
;$
STOP2=   5
;$
LS1A=    6
;$
LS1B=    7
;$
LS2A=    10
;$
LS2B=    11
;$
SRSL1=   12
;$
SRSL2=   13
;$
RTCAN=   14
;$
BLOWC=   16
;$
SRSL3=   15


;$
BLOW=    1000
;$
RTSET=   1001
;$
RAL1=    1002
;$
RAL2=    1003
;$
DV1A=    1004
```

```
;$
DV1B=    1005
;$
DV2A=    1006
;$
DV2B=    1007
;$
RTSL1=   1010


;$
RTSL2=   1011
;$
RTSL3=   1012


;$
RTM1=    1400
;$
RTM2=    1403
;$
RTM3=    1402


;$
TD1=     1600
;$
TD1D=    1601


;$
#
```

```
;START OF MAIN PROGRAM


;$
BLOW=RTSET*/STOPB*/TD1D*(STARB+BLOWC)
1000=1001*/1*/1601*(0+16)

   0 0000 5001 TF       1001
   0 0001 6001 TN       0001
   0 0002 7601 TN       1601
   0 0003 2412 JFN      0012
   0 0004 4000 TF       0000
   0 0005 2010 JFF      0010
   0 0006 4016 TF       0016
   0 0007 2412 JFN      0012
   0 0010 3000 SN       1000
   0 0011 0310 SKP
   0 0012 1000 SF       1000
   0 0013 0000 NOP


;$
RTSET=RTSL1+RTSL2+RTSL3
1001=1010+1011+1012

   0 0016 5010 TF       1010
   0 0017 2024 JFF      0024
   0 0020 5011 TF       1011
   0 0021 2024 JFF      0024
   0 0022 5012 TF       1012
   0 0023 2426 JFN      0026
   0 0024 3001 SN       1001
   0 0025 0010 SKP
   0 0026 1001 SF       1001
   0 0027 0000 NOP


;$
RAL1=(STAR1+RAL1)*/STOP1*/TD1*BLOWC
1002=(2+1002)*/3*/1600*16

   0 0030 4002 TF       0002
   0 0031 2034 JFF      0034
   0 0032 5002 TF       1002
   0 0033 2442 JFN      0042
   0 0034 6003 TN       0003
   0 0035 7600 TN       1600
   0 0036 4016 TF       0016
   0 0037 2442 JFN      0042
```

```
   Ø 0040 3002 SN        1002
   Ø 0041 0010 SKP
   Ø 0042 1002 SF        1002
   Ø 0043 0000 NOP


;$
RAL2=(STAR2+RAL2)*/STOP2*/TD1*BLOWC
1003=(4+1003)*/5*/1600*16

   Ø 0044 4004 TF        0004
   Ø 0045 2050 JFF       0050
   Ø 0046 5003 TF        1003
   Ø 0047 2456 JFN       0056
   Ø 0050 6005 TN        0005
   Ø 0051 7600 TN        1600
   Ø 0052 4016 TF        0016
   Ø 0053 2456 JFN       0056
   Ø 0054 3003 SN        1003
   Ø 0055 0010 SKP
   Ø 0056 1003 SF        1003
   Ø 0057 0000 NOP


;$
DV1A=RTM2+RTM3
1004=1403+1402

   Ø 0060 5403 TF        1403
   Ø 0061 2064 JFF       0064
   Ø 0062 5402 TF        1402
   Ø 0063 2466 JFN       0066
   Ø 0064 3004 SN        1004
   Ø 0065 0010 SKP
   Ø 0066 1004 SF        1004
   Ø 0067 0000 NOP


;$
DV1B=RTM1
1005=1400

   Ø 0072 5400 TF        1400
   Ø 0073 2476 JFN       0076
   Ø 0074 3005 SN        1005
   Ø 0075 0010 SKP
   Ø 0076 1005 SF        1005
   Ø 0077 0000 NOP
```

```
;$
DV2A=RTM3
1006=1402

   0 0102 5402 TF       1402
   0 0103 2506 JFN      0106
   0 0104 3006 SN       1006
   0 0105 0010 SKP
   0 0106 1006 SF       1006
   0 0107 0000 NOP


;$
DV2B=RTM2
1007=1403

   0 0112 5403 TF       1403
   0 0113 2516 JFN      0116
   0 0114 3007 SN       1007
   0 0115 0010 SKP
   0 0116 1007 SF       1007
   0 0117 0000 NOP


;$
RTSL1=RTM1*LS1B*/LS1A
1010=1400*7*/6

   0 0120 5400 TF       1400
   0 0121 4007 TF       0007
   0 0122 6006 TN       0006
   0 0123 2526 JFN      0126
   0 0124 3010 SN       1010
   0 0125 0010 SKP
   0 0126 1010 SF       1010
   0 0127 0000 NOP


;$
RTSL2=RTM2*LS1A*/LS1B*LS2B*/LS2A
1011=1403*6*/7*11*/10

   0 0132 5403 TF       1403
   0 0133 4006 TF       0006
   0 0134 6007 TN       0007
   0 0135 4011 TF       0011
   0 0136 6010 TN       0010
   0 0137 2542 JFN      0142
   0 0140 3011 SN       1011
   0 0141 0010 SKP
```

```
   Ø 0142 1011 SF       1011
   Ø 0143 0000 NOP


;$
RTSL3=RTM3*LS1A*/LS1B*/LS2B*LS2A
1012=1402*6*/7*/11*10

   Ø 0146 5402 TF       1402
   Ø 0147 4006 TF       0006
   Ø 0150 6007 TN       0007
   Ø 0151 6011 TN       0011
   Ø 0152 4010 TF       0010
   Ø 0153 2556 JFN      0156
   Ø 0154 3012 SN       1012
   Ø 0155 0010 SKP
   Ø 0156 1012 SF       1012
   Ø 0157 0000 NOP


;$
RTM1=(SRSL1+RTM1)*/RTM2*/RTM3*/TD1D*(/RTCAN+BLOW)
1400=(12+1400)*/1403*/1402*/1601*(/14+1000)

   Ø 0160 4012 TF       0012
   Ø 0161 2164 JFF      0164
   Ø 0162 5400 TF       1400
   Ø 0163 2576 JFN      0176
   Ø 0164 7403 TN       1403
   Ø 0165 7402 TN       1402
   Ø 0166 7601 TN       1601
   Ø 0167 2576 JFN      0176
   Ø 0170 6014 TN       0014
   Ø 0171 2174 JFF      0174
   Ø 0172 5000 TF       1000
   Ø 0173 2576 JFN      0176
   Ø 0174 3400 SN       1400
   Ø 0175 0010 SKP
   Ø 0176 1400 SF       1400
   Ø 0177 0000 NOP


;$
RTM2=(SRSL2+RTM2)*/RTM1*/RTM3*/TD1D*(/RTCAN+BLOW)
1403=(13+1403)*/1400*/1402*/1601*(/14+1000)

   Ø 0200 4013 TF       0013
   Ø 0201 2204 JFF      0204
   Ø 0202 5403 TF       1403
```

```
   Ø 0203 2616 JFN      0216
   Ø 0204 7400 TN       1400
   Ø 0205 7402 TN       1402
   Ø 0206 7601 TN       1601
   Ø 0207 2616 JFN      0216
   Ø 0210 6014 TN       0014
   Ø 0211 2214 JFF      0214
   Ø 0212 5000 TF       1000
   Ø 0213 2616 JFN      0216
   Ø 0214 3403 SN       1403
   Ø 0215 0010 SKP
   Ø 0216 1403 SF       1403
   Ø 0217 0000 NOP


;$
RTM3=(SRSL3+RTM3)*/RTM1*/RTM2*/TD1D*(/RTCAN+BLOW)
1402=(15+1402)*/1400*/1403*/1601*(/14+1000)

   Ø 0220 4015 TF       0015
   Ø 0221 2224 JFF      0224
   Ø 0222 5402 TF       1402
   Ø 0223 2636 JFN      0236
   Ø 0224 7400 TN       1400
   Ø 0225 7403 TN       1403
   Ø 0226 7601 TN       1601
   Ø 0227 2636 JFN      0236
   Ø 0230 6014 TN       0014
   Ø 0231 2234 JFF      0234
   Ø 0232 5000 TF       1000
   Ø 0233 2636 JFN      0236
   Ø 0234 3402 SN       1402
   Ø 0235 0010 SKP
   Ø 0236 1402 SF       1402
   Ø 0237 0000 NOP


.SEC 10
;$
TD1=(RTCAN*BLOW+TD1)*/TD1D
1600=(14*1000+1600)*/1601

   Ø 0241 4014 TF       0014
   Ø 0242 5000 TF       1000
   Ø 0243 2246 JFF      0246
   Ø 0244 5600 TF       1600
   Ø 0245 2652 JFN      0252
   Ø 0246 7601 TN       1601
   Ø 0247 2652 JFN      0252
```

```
   Ø Ø25Ø 36ØØ SN       16ØØ
   Ø Ø251 ØØ1Ø SKP
   Ø Ø252 16ØØ SF       16ØØ
   Ø Ø253 ØØØØ NOP

.END

ERROR LINES: ØØØØ

LAST LOCATION: Ø Ø253
```

# APPENDIX C

Table C.1

The Microprocessor Program

```
                    ; 8080A   HHJ   4-23-1976
                    ;
                    ;
                    ;--------------------------------------
                    ;SYMBOL TABLE FOR EACH LOGIC ELEMENT
                    ;
                    ;SDK-80 SYSTEM HAS 1024 LOCATIONS FOR
                    ;I/O STATUS,TIMER STATUS AND STACK
                    ;-----------------------------------,--------
1320                STARB   EQU     1320H   ;0PA0
1321                STOPB   EQU     1321H   ;0PA1
1322                STAR1   EQU     1322H   ;0PA2
1323                STOP1   EQU     1323H   ;0PA3
1324                STAR2   EQU     1324H   ;0PA4
1325                STOP2   EQU     1325H   ;0PA5
1326                LS1A    EQU     1326H   ;0PA6
1327                LS1B    EQU     1327H   ;0PA7
                    ;
1328                LS2A    EQU     1328H   ;0PB0
1329                LS2B    EQU     1329H   ;0PB1
132A                SRSL1   EQU     132AH   ;0PB2
132B                SRSL2   EQU     132BH   ;0PB3
132C                RTCAN   EQU     132CH   ;0PB4
132D                SRSL3   EQU     132DH   ;0PB5
132E                BLOWC   EQU     132EH   ;0PB6
                    ;0PB7 TO 0PC7 SPARE INPUTS
                    ;
                    ;END OF INPORT TABLE
```

Table C.1 (continued)

```
                        ;OUTPUTS START AT 1340H
                        ;
1340                    BLOW    EQU     1340H   ;1PA0
1341                    RTSET   EQU     1341H   ;1PA1
1342                    RAL1    EQU     1342H   ;1PA2
1243                    RAL2    EQU     1243H   ;1PA3
1344                    DV1A    EQU     1344H   ;1PA4
1345                    DV1B    EQU     1345H   ;1PA5
1346                    DV2A    EQU     1346H   ;1PA6
1347                    DV2B    EQU     1347H   ;1PA7
                        ;
1348                    RTSL1   EQU     1348H   ;1PB0
1349                    RTSL2   EQU     1349H   ;1PB1
134A                    RTSL3   EQU     134AH   ;1PB2
134B                    CHECK   EQU     134BH   ;1PB3
134C                    RTM1    EQU     134CH   ;1PB4
134D                    RTM2    EQU     134DH   ;1PB5
134E                    RTM3    EQU     134EH   ;1PB6
                                                ;
                        ;
1350                    TD1     EQU     1350H   ;1PC0 TIME AT 1380H
1351                    TD1D    EQU     1351H   ;1PC1
                        ;
                        ;
                        ;TIMER PRESET TABLE STARTS AT 1380H
                        ;
1380                    TD1P    EQU     1380H   ;10SEC
                        ;
                        ;END OF SYMBOL TABLE
0031                    TABLE   EQU     TD1D-STARB      ;LENGTH OF TABLE
                        ;
                        ;
```

Table C.1 (continued)

```
0000                    ORG     00H     ;PROGRAM COUNTER AT POWER UP
0000 C30E01             JMP     START   ;START OF PROGRAM
                        ;---------------------------------------
                        ;INTERUPT SUBROUTINE
                        ;
                        ;STORES CPU AND REGISTURE STATUS
                        ;CALLS EACH TIMER EVER 0.1 SECOND
                        ;RESTORES CPU AND REGISTURE STARUS
                        ;RETURNS TO MAIN PROGRAM
                        ;---------------------------------------
0003                    ORG     38H     ;HARDWARE INTERRUPT LOCATON
0038 C5         INTR:   PUSH    B       ;SAVE B
0039 D5                 PUSH    D       ;SAVE D
003A E5                 PUSH    H       ;SAVE H
003B F5                 PUSH    PSW     ;SAVE PSW
003C CD4500             CALL    T1      ;CALL ALL TIMER ROUTINES
003F F1                 POP     PSW     ;RESTORE PSW
0040 E1                 POP     H       ;RESTORE H
0041 D1                 POP     D       ;RESTORE D
0042 C1                 POP     B       ;RESTORE B
0043 FB                 EI              ;ENABLE INTERRUPTS
0044 C9                 RET             ;RETURN TO MAIN PROGRAM
                        ;
0045 3A5013     T1:     LDA     TD1     ;GET STATUS OF TD1
0048 FE00               CPI     00H     ;CHECK FOR TIMED OUT
004A CA5900             JZ      $+15    ;JUMP IF OFF    TO RESET TD1
004D 3A8013             LDA     TD1P    ;OTHERWISE GET REMANING TIME
0050 FE00               CPI     00H     ;SEE IF TIMED OUT
0052 CA5B00             JZ      $+9     ;JUMP TO STOR IF TIMED OUT
0055 3D                 DCR     A       ;OTHERWISE DECERMENT TD1 0.1 SEC
0056 C35B00             JMP     $+5     ;JUMP TO STORE TIME
0059 3E64               MVI     A,100   ;OTHERWISE RESET TD1 TO 10 SEC
005B 328013             STA     TD1P    ;STORE REMANING TIME
005E C9                 RET
```

Table C.1 (continued)

```
                        ;-----------------------------------------
                        ;THE FOLLOWING SUBROUTINE MASKS AN INPORT
                        ;ONE BIT AT A TIME AND STORES IN A
                        ;TABLE IN RAM
                        ;
                        ;-----------------------------------------
                        ;
005F 47         IPTOM:  MOV     B,A     ;SAVE PORT STATUS
0060 E601               ANI     01H     ;MASK BIT 0
0062 0F                 RRC             ;ROTATE BIT 0 TO D7
0063 CD9930             CALL    MOVE    ;MOVE BIT 0 TO RAM
0066 E602               ANI     02H     ;MASK BIT 1
0068 0F                 RRC
0069 0F                 RRC             ;ROTATE BIT 1 TO D7
006A CD9900             CALL    MOVE    ;MOVE BIT 1 TO RAM
006D E604               ANI     04H     ;MASK BIT 2
006F 0F                 RRC
0070 0F                 RRC
0071 0F                 RRC             ;ROTATE BIT 2 TO D7
0072 CD9900             CALL    MOVE    ;MOVE BIT 2 TO RAM
0075 E608               ANI     08H     ;MASK BIT 3
0077 0F                 RRC
0078 0F                 RRC
0079 0F                 RRC
007A 0F                 RRC             ;ROTATE BIT 3 TO D7
007B CD9900             CALL    MOVE    ;MOVE BIT 3 TO RAM
007E E610               ANI     10H     ;MASK BIT 4
0080 07                 RLC
0081 07                 RLC
0082 07                 RLC             ;ROTATE BIT 4 TO D7
0083 CD9900             CALL    MOVE    ;MOVE BIT 4 TO RAM
0086 E620               ANI     20H     ;MASK BIT 5
```

Table C.1 (continued)

```
0088 07                 RLC
0089 07                 RLC             ;ROTATE BIT 5 TO D7
008A CD9900             CALL    MOVE    ;MOVE BIT 5 TO RAM
008D E640               ANI     40H     ;MASK BIT 6
008F 07                 RLC             ;ROTATE BIT 6 TO D7
0090 CD9900             CALL    MOVE    ;MOVE BIT 6 TO RAM
0093 E680               ANI     80H     ;MASK BIT 7
0095 CD9900             CALL    MOVE    ;MOVE BIT 7 TO RAM
0098 C9                 RET             ;RETURN TO MAIN PROGRAM
                        ;
                        ;-----------------------------------------------
                        ;SUBROUTINE MOVES DATA FROM ACC TO MEMORY
                        ;AND INCERMENTS H AND L REG. FOR
                        ;NEXT LOCATION IN RAM TABLE
                        ;
                        ;-----------------------------------------------
                        ;
0099 77         MOVE:   MOV     M,A     ;STORE IN RAM
009A 23                 INX     H       ;INCERMENT ONE POS IN TABLE
009B 78                 MOV     A,B     ;RESTORD PORT STATUS
009C C9                 RET             ;RETURN TO MAIN PROGRAM
                        ;
                        ;-----------------------------------------------
                        ;SUBROUTINES TO MOVE THE RESULTS OF AN
                        ;EQUATION TO RAM TABLE AND BUILD A OUTPORT
                        ;BYTE TO SEND OUT AFTER "07" HAS BEEN CALLED
                        ;
                        ;---------------,-------------------------------
                        ;
009D E680       00:     ANI     80H     ;MASK BIT 7
009F 77                 MOV     M,A     ;MOVE BIT 0 TO RAM
00A0 23                 INX     H       ;POINT TO NEXT RAM LOCATION
```

Table C.1 (continued)

```
00A1 07              RLC                 ;ROTATE TO D0
00A2 320013          STA     1300H       ;BIT 0 OF OUTPORT BYTE
00A5 C9              RET                 ;RETURN TO MAIN PROGRAM
00A6 E680      01:   ANI     80H         ;MASK BIT 7
00A8 77              MOV     M,A         ;MOVE BIT 1 TO RAM
00A9 23              INX     H           ;POINT TO NEXT RAM LOCATION
00AA 07              RLC                 ;ROTATE TO D1
00AB 07              RLC
00AC 47              MOV     B,A         ;MOVE BIT 1 TO B
00AD 3A0013          LDA     1300H       ;STATUS OF PREVIOUS BIT
00B0 B0              ORA     B           ;OR WITH BIT 1
00B1 320013          STA     1300H       ;STORE BIT 1 AND BIT 0
00B4 C9              RET                 ;RETURN TO MAIN PROGRAM
00B5 E680      02:   ANI     80H         ;MASK BIT 7
00B7 77              MOV     M,A         ;MOVE BIT 1 TO RAM
00B8 23              INX     H           ;POINT TO NEXT RAM LOCATION
00B9 07              RLC
00BA 07              RLC                 ;ROTATE TO D2
00BB 07              RLC
00BC 47              MOV     B,A         ;MOVE BIT 2 TO B
00BD 3A0013          LDA     1300H       ;STATUS OF PREVIOUS BITS
00C0 B0              ORA     B           ;OR BIT WITH PREVIOUS BITS
00C1 320013          STA     1300H       ;STORE BIT 0 THRU BIT 2
00C4 C9              RET                 ;RETURN TO MAIN PROGRAM
00C5 E680      03:   ANI     80H         ;MASK BIT 7
00C7 77              MOV     M,A         ;MOVE BIT 3 TO RAM
00C8 23              INX     H           ;POINT TO NEXT RAM LOCATION
00C9 07              RLC
00CA 07              RLC                 ;ROTATE TO D3
00CB 07              RLC
00CC 07              RLC
00CD 47              MOV     B,A         ;MOVE BIT 3 TO B
```

Table C.1 (continued)

```
00CE 3A0013          LDA     1300H   ;STATUS OF PREVIOUS BITS
00D1 B0              ORA     B       ;OR BIT 3 WITH PREVIOUS BITS
00D2 320013          STA     1300H   ;STORE BIT 0 THRU BIT 3
00D5 C9              RET             ;RETURN TO MAIN PROGRAM
00D6 E680      04:   ANI     80H     ;MASK D7
00D8 77              MOV     M,A     ;MOVE BIT 4 TO RAM
00D9 23              INX     H       ;POINT TO NEXT RAM LOCATION
00DA 0F              RRC             ;ROTATE TO D4
00DB 0F              RRC
00DC 0F              RRC
00DD 47              MOV     B,A     ;MOVE BIT 4 TO B
00DE 3A0013          LDA     1300H   ;STATUS OF PREVIOUS BITS
00E1 B0              ORA     B       ;OR BIT 4 WITH PREVIOUS BITS
00E2 320013          STA     1300H   ;STORE BIT 0 THRU BIT 4
00E5 C9              RET             ;RETURN TO MAIN PROGRAM
00E6 E680      05:   ANI     80H     ;MASK BIT 7
00E8 77              MOV     M,A     ;MOVE BIT 5 TO RAM
00E9 23              INX     H       ;POINT TO NEXT RAM LOCATION
00EA 0F              RRC             ;ROTATE TO D5
00EB 0F              RRC
00EC 47              MOV     B,A     ;MOVE BIT 5 TO B
00ED 3A0013          LDA     1300H   ;STATUS OF PREVIOUS BITS
00F0 B0              ORA     B       ;OR BIT 5 WITH PREVINUS BITS
00F1 320013          STA     1300H   ;STORE BIT 0 THRU BIT 5
00F4 C9              RET             ;RETURN TO MAIN PROGRAM
00F5 E680      06:   ANI     80H     ;MASK BIT 7
00F7 77              MOV     M,A     ;MOVE BIT 6 TO RAM
00F8 23              INX     H       ;POINT TO NEXT RAM LOCATION
00F9 0F              RRC             ;ROTATE TO D6
00FA 47              MOV     B,A     ;MOVE BIT 6 TO B
00FB 3A0013          LDA     1300H   ;STATUS OF PREVIOUS BITS
00FE B0              ORA     B       ;OR BIT 6 WITH PREVIOUS BITS
00FF 320013          STA     1300H   ;STORE BIT 0 THRU BIT 6
```

Table C.1 (continued)

```
0102 C9                  RET                ;RETTRN TO MAIN PROGRAM
0103 E680      07:       ANI      80H       ;MASK BIT 7
0105 77                  MOV      M,A       ;MOVE BIT 7 TO RAM
0106 23                  INX      H         ;POINT TO NEXT RAM LOCATION
0107 47                  MOV      B,A       ;MOVE BIT 7 TO B
0108 3A0013              LDA      1300H     ;STATUS OF PREVIOUS BITS
010B B0                  ORA      B         ;OR BIT 7 WITH PREVIOUS BITS
010C 2F                  CMA                ;COMPLEMENT OUTPUT BYTE BECAUSE
                                            ;THE OUTPUT AMPLIFERS USED IN
                                            ;THIS SYSTEM ARE INVERTERS
010D C9                  RET                ;RETURN TO MAIN PROGRAM WITH THE
                                            ;OUTPUT BYTE IN REGISTURE A
010E 31FC13    START:    LXI      SP,13FCH          ;SET TOP OF STACK
                         ;
                         ;
                         ;-----------------------------------------------
                         ;INITILIZE THE RAM TABLE ON POWER UP
                         ;-----------------------------------------------
                         ;
                         ;
0111 212013              LXI      H,STARB ;START OF TABLE
0114 0631                MVI      B,TABLE ;LENGTH COUNTER
0116 3F00                MVI      A,00H   ;LOAD 00 IN ACC
0118 77        INIT:     MOV      M,A     ;LOADS 00 IN ALL LOCATIONS
0119 23                  INX      H       ;MOVE TO NEXT BYTE
011A 05                  DCR      B       ;DECERMENT COUNTER
011B C21801              JNZ      INIT    ;DO NEXT LOC IF B IS NOT 0
011E 3F80                MVI      A,80H   ;CONT. WORD 0 CHIP 1
0120 D3EF                OUT      0EFH    ;SEND TO CHIP 1
                         ;
                         ;
                         ;-----------------------------------------------
                         ;PROGRAM GETS STATUS OF ALL INPUTS AND
```

Table C.1 (continued)

```
                        ;STORES IN RAM.
                        ;SOLVES THE BOOLEAN EQUATION FOR
                        ;EACH OUTPUT AND STORES THE STATUS
                        ;IN RAM.        SENDS A BYTE OUT
                        ;TO THE CORRECT PORT AFTER EACH
                        ;D7 EQUATION
                        ;
                        ;FOR THIS PROGRAM:      LOGICAL NOT=/
                        ;                               AND=*
                        ;                               OR=+
                        ;
                        ;-----------------------------------------
0122 212013   LOOP:     LXI     H,STARB ;START OF INPORT TABLE
0125 DBF4               IN      0F4H    ;CHIP 0 PORT A STATUS
0127 CD5F00             CALL    IPTOM   ;SEND TO RAM TABLE
012A DBF5               IN      0F5H    ;CHIP 0 PORT B STATUS
012C CD5F00             CALL    IPTOM   ;SEND TO RAM TABLE
012F DBF6               IN      0F6H    ;CHIP 0 PORT C STATUS
0131 CD5F00             CALL    IPTOM   ;SEND TO RAM TABLE
0134 214013   SIX:      LXI     H,BLOW  ;START OF OUTPUT TABLE
0137 FB                 EI              ;ENABLE INTERRUPTS
                        ;
                        ;
                        ;-----------------------------------------
                        ;START OF MAIN PROGRAM FOR SOLVING
                        ;BOOLEAN EQUATIONS
                        ;-----------------------------------------
                        ;---------------------------------
                        ;BOOLEAN EXPRESSION FOR BLOWER CONTROL
                        ;IN SYMBOLIC NOTATION
                        ;
                        ;BLOW=RTSET*/STOPB*/TD1D*(STARB+BLOWC)
                        ;---------------------------------
```

Table C.1 (continued)

```
                        ;
                        ;
0138 3A4113             LDA     RTSET   ;LOAD STATUS OF RTSET IN A
013B 47                 MOV     B,A     ;MOVE STATUS OF RTSET TO B
013C 3A2113             LDA     STOPB   ;STATUS OF STOPB IN A
013F 2F                 CMA             ;LOGICAL NOT OF STOPB
0140 4F                 MOV     C,A     ;MOVE STATUS OF /STOTB TO C
0141 3A5113             LDA     TD1D    ;STATUS OF TD1D TO A
0144 2F                 CMA             ;LOGICAL NOT OF TD1D
0145 A0                 ANA     B       ;RTSET*(/TD1D)
0146 A1                 ANA     C       ;/STOPB*(RTSET*/TD1D)
0147 57                 MOV     D,A     ;SAVE (RTSET*/STOPB*/TD1D) IN D
0148 3A2013             LDA     STARB   ;LOAD STATUS OF STARB IN A
014B 47                 MOV     B,A     ;SAVE STARB IN B
014C 3A2E13             LDA     BLOWC   ;LOAD STATUS OF BLOWC IN A
014F B0                 ORA     B       ;STARB+(BLOWC)
0150 A2                 ANA     D       ;(STARB+BLOWC)*RTSER*/STOPB*/TD1
0151 CD9D00             CALL    00      ;CALL SUBROUTINE 00 TO STORE
                                        ;STATUS OF BLOW IN RAM
                        ;
                        ;
                        ;RTSET=RTSL1+RTSL2+RTSL3
0154 3A4813             LDA     RTSL1
0157 47                 MOV     B,A
0158 3A4913             LDA     RTSL2
015B 4F                 MOV     C,A
015C 3A4A13             LDA     RTSL3
015F B0                 ORA     B
0160 B1                 ORA     C
0161 CDA600             CALL    01
                        ;
                        ;
                        ;RAL1=(STAR1+RAL1)*/STOP1*/TD1*BLOWC
```

Table C.1 (continued)

```
0164 3A2213          LDA     STAR1
0167 47              MOV     B,A
0168 3A4213          LDA     RAL1
016B B0              ORA     B
016C 57              MOV     D,A
016D 3A2313          LDA     STOP1
0170 2F              CMA
0171 47              MOV     B,A
0172 3A5013          LDA     TD1
0175 2F              CMA
0176 4F              MOV     C,A
0177 3A2F13          LDA     BLOWC
017A A0              ANA     B
017B A1              ANA     C
017C A2              ANA     D
017D CDB500          CALL    02
                     ;
                     ;
                     ;RAL2=(STAR2+RAL2)*/STOP2*/TD1*BLOWC
0180 3A2413          LDA     STAR2
0183 47              MOV     B,A
0184 3A4312          LDA     RAL2
0187 B0              ORA     B
0188 57              MOV     D,A
0189 3A2513          LDA     STOP2
018C 2F              CMA
018D 47              MOV     B,A
018E 3A5013          LDA     TD1
0191 2F              CMA
0192 4F              MOV     C,A
0193 3A2E13          LDA     BLOWC
0196 A0              ANA     B
0197 A1              ANA     C
```

Table C.1 (continued)

```
0198 A2          ANA     D
0199 CDC500      CALL    03
                 ;
                 ;
                 ;DV1A=RTM2+RTM3
019C 3A4D13      LDA     RTM2
019F 47          MOV     B,A
01A0 3A4E13      LDA     RTM3
01A3 B0          ORA     B
01A4 CDD600      CALL    04
                 ;
                 ;
                 ;DV1B=RTM1
01A7 3A4C13      LDA     RTM1
01AA CDE600      CALL    05
                 ;
                 ;
                 ;DV2A=RTM3
01AD 3A4E13      LDA     RTM3
01B0 CDF500      CALL    06
                 ;DV2B=RTM2
01B3 3A4D13      LDA     RTM2
01B6 CD0301      CALL    07
01B9 D3EC        OUT     0ECH    ;BYTE TO CHIP 1 PORT A
                 ;
                 ;
                 ;RTSL1=RTM1*LS1B*/LS1A
01BB 3A4C13      LDA     RTM1
01BE 47          MOV     B,A
01BF 3A2713      LDA     LS1B
01C2 A0          ANA     B
01C3 47          MOV     B,A
01C4 3A2613      LDA     LS1A
```

Table C.1 (continued)

```
01C7 2F          CMA
01C8 A0          ANA     B
01C9 CD9D00      CALL    00
                 ;
                 ;
                 ;RTSL2=RTM2*LS1A*/LS1B*LS2B*/LS2A
01CC 3A4D13      LDA     RTM2
01CF 47          MOV     B,A
01D0 3A2613      LDA     LS1A
01D3 A0          ANA     B
01D4 47          MOV     B,A
01D5 3A2713      LDA     LS1B
01D8 2F          CMA
01D9 A0          ANA     B
01DA 47          MOV     B,A
01DB 3A2913      LDA     LS2B
01DE A0          ANA     B
01DF 47          MOV     B,A
01E0 3A2813      LDA     LS2A
01E3 2F          CMA
01E4 A0          ANA     B
01E5 CDA600      CALL    01
                 ;
                 ;
                 ;RTSL3=RTM3*LS1A*/LS1B*/LS2B*LS2A
01E8 3A4E13      LDA     RTM3
01EB 47          MOV     B,A
01EC 3A2613      LDA     LS1A
01EF A0          ANA     B
01F0 47          MOV     B,A
01F1 3A2713      LDA     LS1B
01F4 2F          CMA
01F5 A0          ANA     B
```

Table C.1 (continued)

```
01F6 47          MOV     B,A
01F7 3A2913      LDA     LS2B
01FA 2F          CMA
01FB A0          ANA     B
01FC 47          MOV     B,A
01FD 3A2813      LDA     LS2A
0200 A0          ANA     B
0201 CDB500      CALL    02
                 ;
                 ;
                 ;CHECK=/CHECK
                 ;
0204 3A4B13      LDA     CHECK
0207 2F          CMA
0208 CDC500      CALL    03
                 ;
                 ;
                 ;RTM1=(SRSL1+RTM1)*/RTM2*/RTM3*
                 ;       /TD1D*(/RTCAN+BLOW)
020B 3A2A13      LDA     SRSL1
020E 47          MOV     B,A
020F 3A4C13      LDA     RTM1
0212 B0          ORA     B
0213 57          MOV     D,A
0214 3A4D13      LDA     RTM2
0217 2F          CMA
0218 47          MOV     B,A
0219 3A4E13      LDA     RTM3
021C 2F          CMA
021D 4F          MOV     C,A
021E 3A5113      LDA     TD1D
0221 2F          CMA
```

Table C.1 (continued)

```
0222 A0          ANA     B
0223 A1          ANA     C
0224 A2          ANA     D
0225 57          MOV     D,A
0226 3A2C13      LDA     RTCAN
0229 2F          CMA
022A 47          MOV     B,A
022B 3A4013      LDA     BLOW
022F B0          ORA     B
022F A2          ANA     D
0230 CDD600      CALL    04
                 ;
                 ;
                 ;RTM2=(SRSL2+RTM2)*/RTM1*/RTM3
                 ;       */TD1D*(/RTCAN+BLOW)
0233 3A2B13      LDA     SRSL2
0236 47          MOV     B,A
0237 3A4D13      LDA     RTM2
023A B0          ORA     B
023B 57          MOV     D,A
023C 3A4C13      LDA     RTM1
023F 2F          CMA
0240 47          MOV     B,A
0241 3A4E13      LDA     RTM3
0244 2F          CMA
0245 4F          MOV     C,A
0246 3A5113      LDA     TD1D
0249 2F          CMA
024A A0          ANA     B
024B A1          ANA     C
024C A2          ANA     D
```

Table C.1 (continued)

```
024D 57          MOV     D,A
024E 3A2C13      LDA     RTCAN
0251 2F          CMA
0252 47          MOV     B,A
0253 3A4A13      LDA     BLOW
0256 B0          ORA     B
0257 A2          ANA     D
0258 CDE600      CALL    05
                 ;
                 ;
                 ;RTM3=(SRSL3+RTM3)*/RTM1*/RTM2*
                 ;       /TD1D*(/RTCAN+BLOW)
025B 3A2D13      LDA     SRSL3
025E 47          MOV     B,A
025F 3A4E13      LDA     RTM3
0262 B0          ORA     B
0263 57          MOV     D,A
0264 3A4C13      LDA     RTM1
0267 2F          CMA
0268 47          MOV     B,A
0269 3A4D13      LDA     RTM2
026C 2F          CMA
026D 4F          MOV     C,A
026E 3A5113      LDA     TD1D
0271 2F          CMA
0272 A0          ANA     B
0273 A1          ANA     C
0274 A2          ANA     D
0275 57          MOV     D,A
0276 3A2C13      LDA     RTCAN
0279 2F          CMA
027A 47          MOV     B,A
```

Table C.1 (continued)

```
027B 3A4013          LDA     BLOW
027E B0              ORA     B
027F A2              ANA     D
0280 CDF500          CALL    06
                     ;
                     ;
                     ;IPB7 SPARE
0283 3E00            MVI     A,00H
0285 CD0301          CALL    07
0288 D3ED            OUT     0EDH     ;BYTE TO CHIP 1 PORT B
                     ;
                     ;
                     ;TD1=(RTCAN*BLOW+TD1)*/TD1D
028A 3A2C13          LDA     RTCAN
028D 47              MOV     B,A
028E 3A4013          LDA     BLOW
0291 A0              ANA     B
0292 47              MOV     B,A
0293 3A5013          LDA     TD1
0296 B0              ORA     B
0297 57              MOV     D,A
0298 3A5113          LDA     TD1D
029B 2F              CMA
029C A2              ANA     D
029D CD9D00          CALL    09
                     ;
                     ;
                     ;TD1D=(0SEC AT TD1P)*TD1
02A0 3A8013          LDA     TD1P     ;GET STATUS OF TIME
02A3 FE00            CPI     00H      ;CHECK FOR OFF
02A5 CAAD02          JZ      $+8      ;JMP IF TIMED OUT
02A8 3E00            MVI     A,00H    ;SET OFF
```

Table C.1 (continued)

```
02AA C3B002          JMP     $+6       ;JUMP TO CALL 01
02AD 3A5013          LDA     TD1       ;STORE STATUS OF TD1D
02B0 CDA600          CALL    01        ;STOR IN RAM TABLE
                     ;
                     ;
02B3 3E00            MVI     A,00H     ;SPARE PORTS=0
02B5 CDB500          CALL    02
02B8 CDC500          CALL    03
02BB CDD600          CALL    04
02BF CDE600          CALL    05
02C1 CDF500          CALL    06
02C4 CD0301          CALL    07
02C7 D3EE            OUT     0EEH      ;BYTE TO CHIP 1 PORT C
02C9 C32201          JMP     LOOP
0000                 END
```

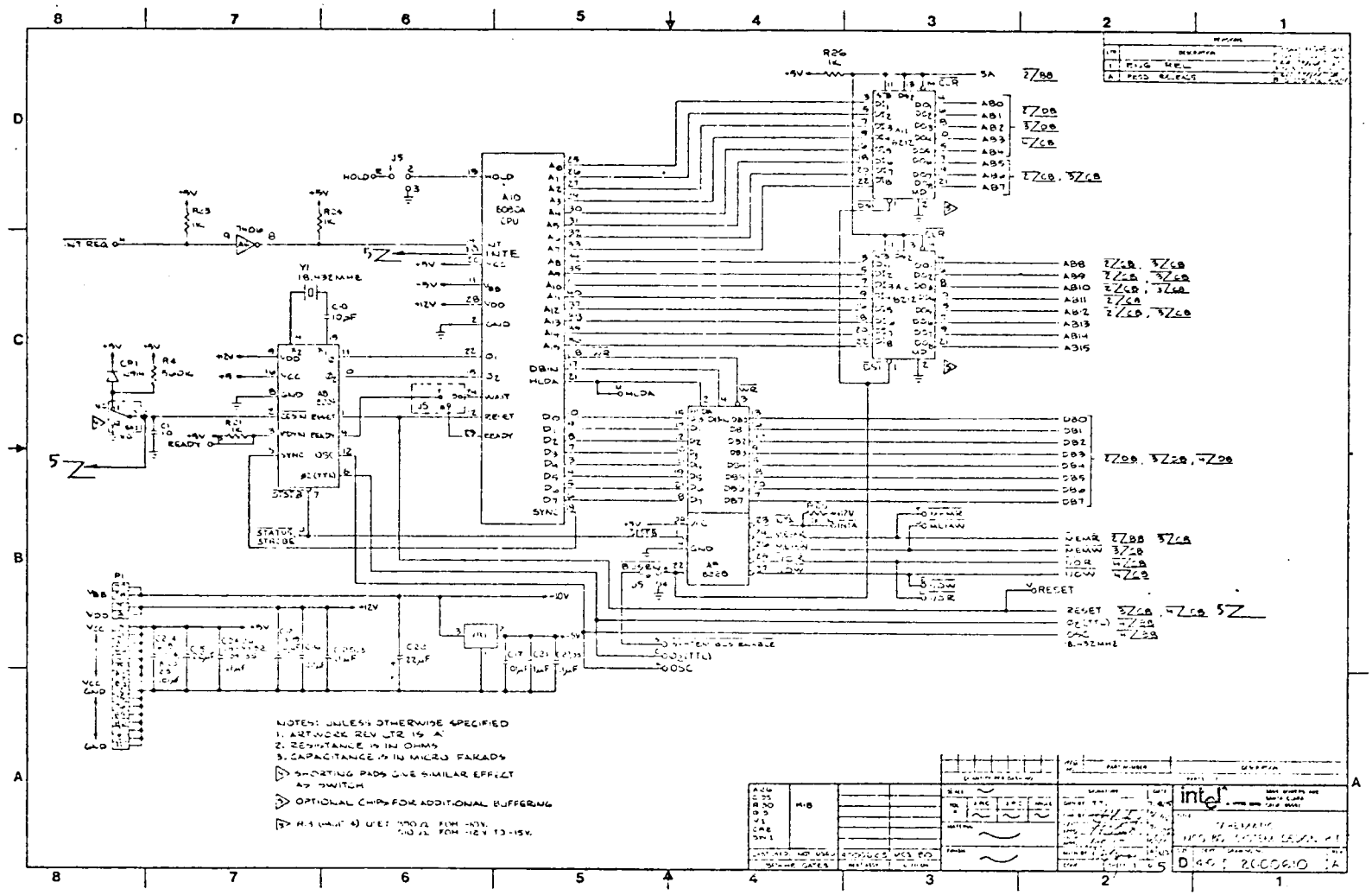


Figure C.1. Schematic MCS-80 System Design Kit.

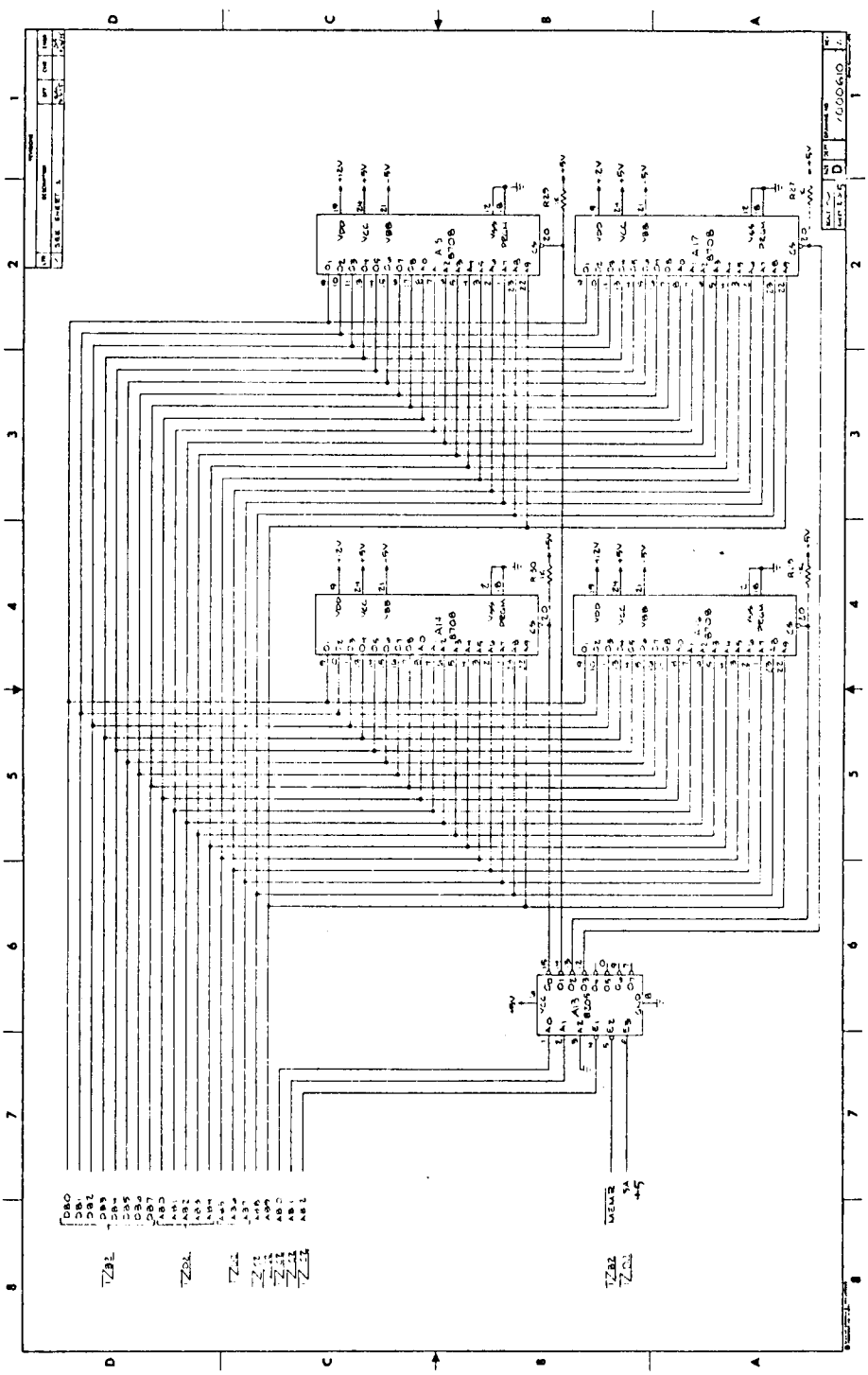


Figure C.1 (continued)

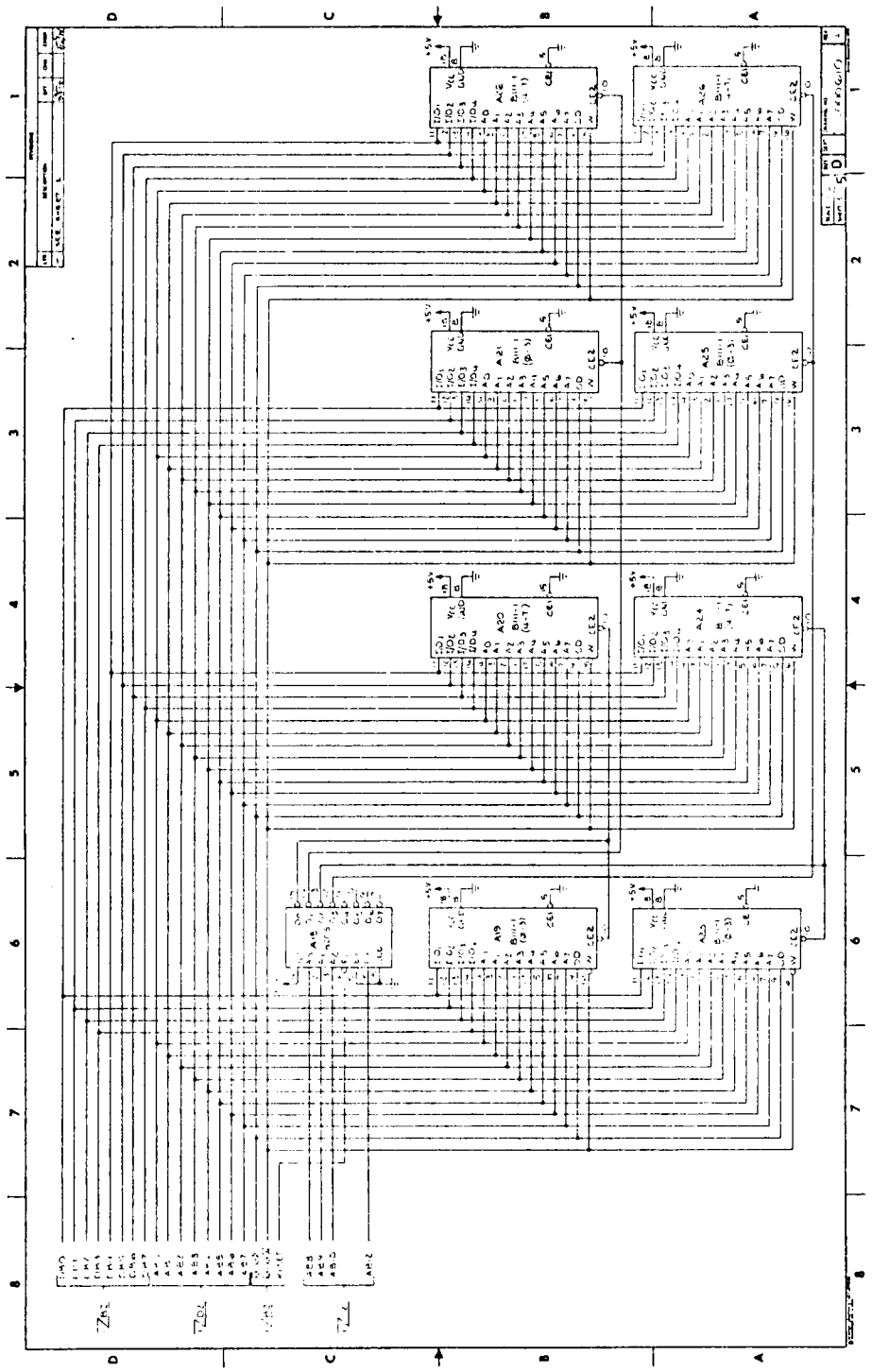


Figure C.1 (continued)

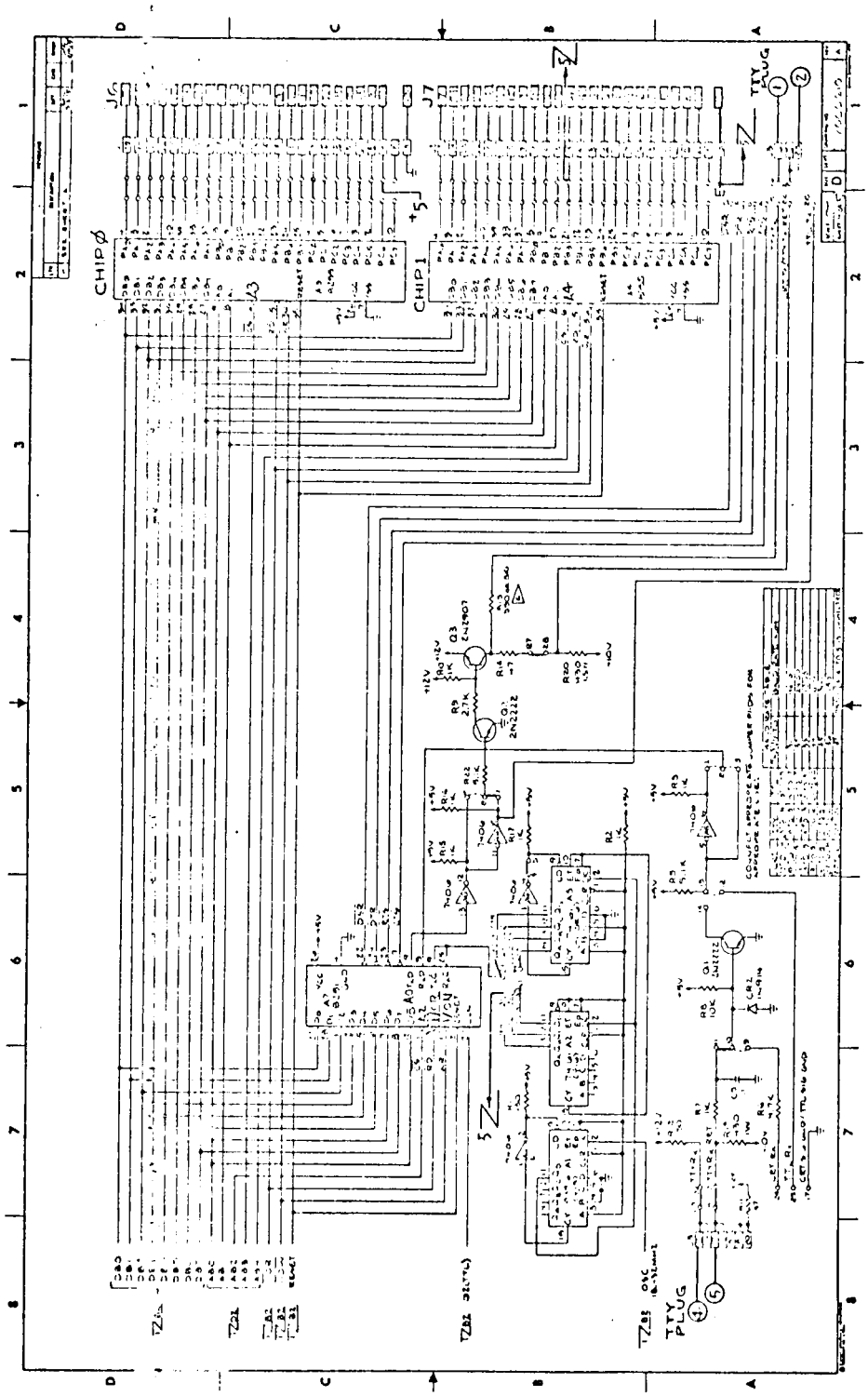



Figure C.1 (continued)

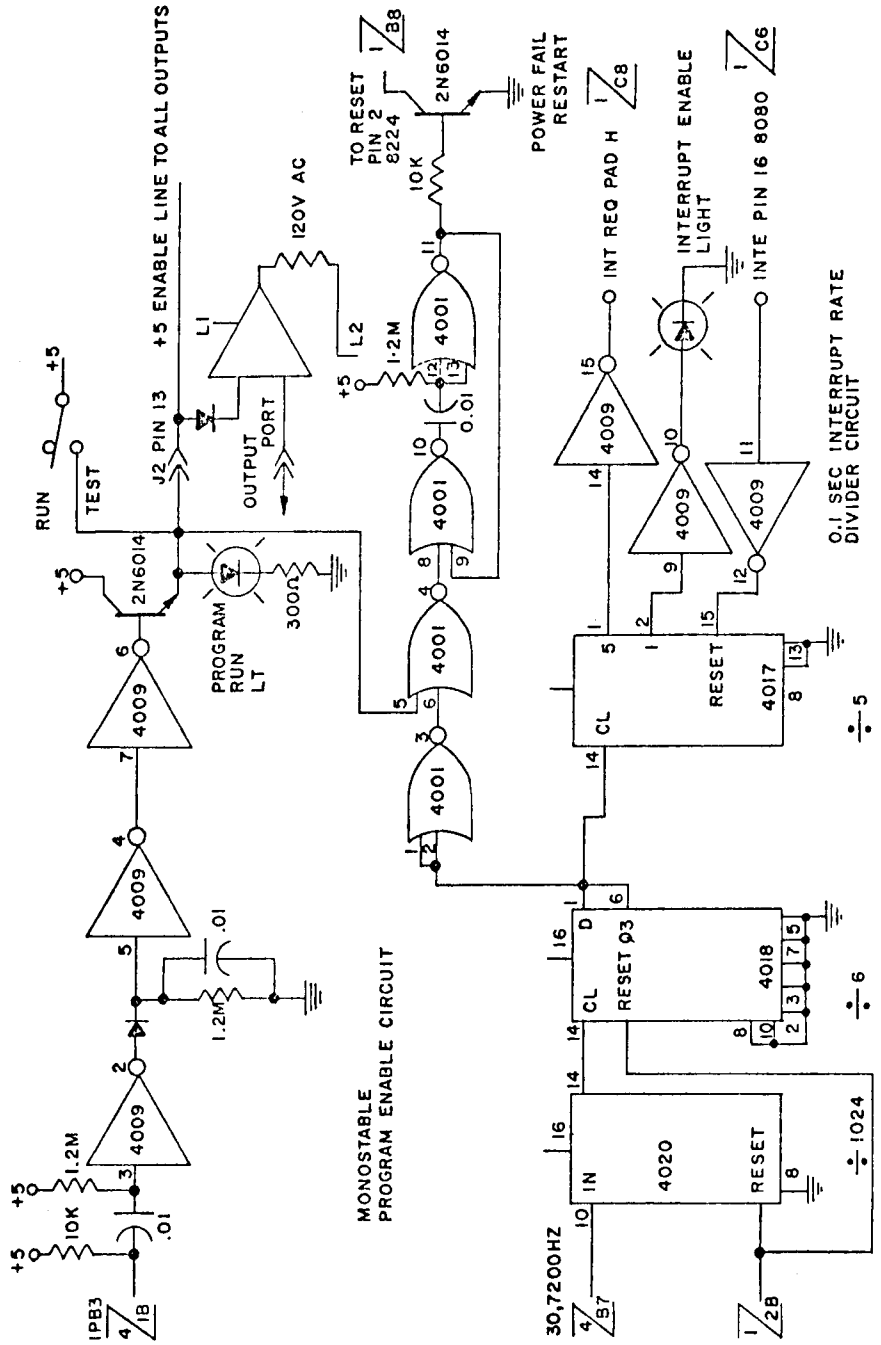



Figure C.1 (continued)

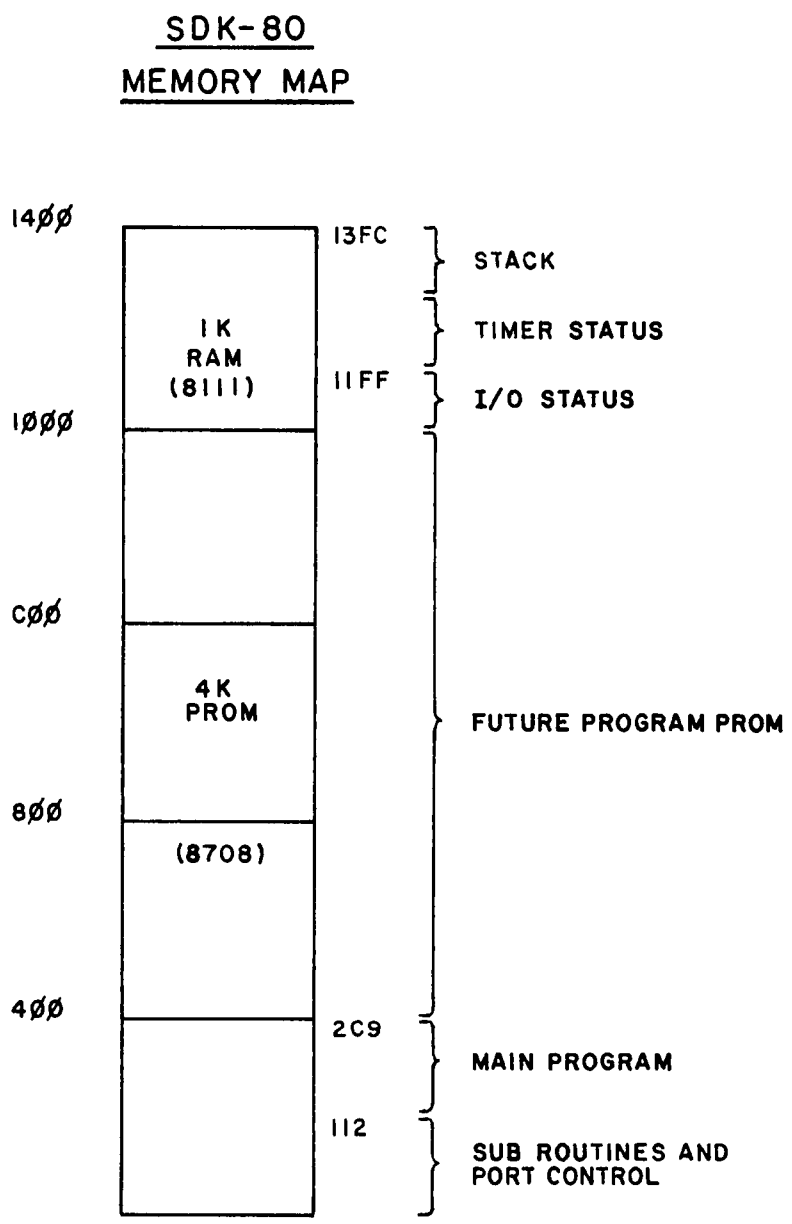



Figure C.2. SDK-80 Memory Map.

# VITA

Harold Hancil Jones was born in Greeneville, Tennessee, February 15, 1941. He attended elementary school in Church Hill, Tennessee, and was graduated from Church Hill High School in 1959. He entered the University of Tennessee in 1960. He graduated with a Bachelor of Science degree in Electrical Engineering in March 1966 and has been employed by Tennessee Eastman Company since April 1966. At Tennessee Eastman Company his job responsibilities have included process instrumentation, power system distribution and generation and process control by digital logic. He currently is a Senior Electrical Engineer in the Engineering division.