To the Graduate Council:

I am submitting a thesis written by Rashid Naseer Khan entitled "A Distress Alarm Keyer." 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.

R. W. Rochelle

R. W. Rochelle, Major Professor

We have read this thesis
and recommend its acceptance:

Marshall Pace

Donald W. Bouldin

Accepted for the Council:

L. Evans Peck

Vice Chancellor
Graduate Studies and Research

A DISTRESS ALARM KEYER

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Rashid Naseer Khan

August, 1979

ACKNOWLEDGEMENTS

The author wishes to express his profound thanks and deep appreciation to Dr. R. W. Rochelle for his extensive encouragement, valuable assistance and general helpfulness throughout the course of this work. Special thanks are due to Dr. D. W. Bouldin and Dr. M. Pace for going through the thesis and assuming the responsibility as committee members.

The author is deeply indebted to his family for their patience and encouragement from thousands of miles away. Appreciation is also extended to Mrs. Denise Smiddy for her skillful service in typing the thesis.

ABSTRACT

The National Aeronautics and Space Administration has proposed a Satellite-Aided Search and Rescue system concept. In the event of an emergency a distress message can be transmitted to the Search and Rescue Center via a satellite. The distress message can be entered and stored in a Distress Alarm Keyer in binary-coded-decimal format. The Distress Alarm Keyer has been designed and implemented using Intel's MCS-48 microcomputer family. The flexibility of the microprocessor-based design of the Keyer allows the expansion of its functions without any major modifications. A comparison has been made between the present software design and the hard-wired version of the Distress Alarm Keyer which was developed in 1976.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Earth satellites are actively engaged in improving man's existence. They are presently employed in providing communication to previously inaccessible portions of our globe. They provide daily information useful in weather predictions and give a means for issuing advance warnings for severe weather conditions. Satellites are also employed in monitoring the resources of the earth. Future satellite systems will control the flow of air traffic over the oceans and the ships at sea.

Another important service could be the use of satellites in Search and Rescue (SAR) of aircraft, ships, spacecraft and individuals in distress. The National Aeronautics and Space Administration (NASA) has proposed a Satellite-Aided Search and Rescue system concept. The system envisages a Data Collection Platform (DCP) capable of transmitting information to the orbiting spacecraft. A distress message can be stored in a Distress Alarm Keyer in binary-coded-decimal format. The distress message will contain information regarding the nature of emergency, identification data, estimated position, weather conditions etc. The Distress Alarm Keyer is capable of communicating with the DCP. Whenever a request for a message is received by the Distress Alarm Keyer, it would be sent to the DCP for onward transmission to the Satellite. The satellite will relay the message to an earth station which will detect the signals and process the information received to determine the position location. The data will be relayed to the rescue coordination center where the SAR forces will be alerted and deployed. The position location

capability of the satellite-aided system will give a distress location with an accuracy of about 5-10 Km.

One of the important components of this system is the Distress Alarm Keyer. A hard-wired logic design of the Distress Alarm Keyer was realized in 1976 by Dale Rickard and Paul Satterlee, Jr. Afterwards a modification in the circuit was carried out by Dale Rickard and Frank Kerr, III in 1977. Now, a software version of the Distress Alarm Keyer has been developed using Intel's MCS-48 microprocessor family. It is capable of storing a message entered from the keyboard and transmits it to the DCP whenever a request for the message is received. Chapter II describes the development of the hard-wired version of the Distress Alarm Keyer. The functions and the operation of the microprocessor version are also included in this chapter.

The advantages of using a microprocessor to develop a system are: quick design cycles, small physical size, low costs and flexibility of design. The logic system design becomes simply the manipulation of functional blocks. The hardware development of the Distress Alarm Keyer using a microprocessor is described in Chapter III.

In the microprocessor-based system the control sequence is stored and executed as a software program. The Read-Only-Memory (ROM) replaces a large number of integrated circuits used in hard-wired logic designs. Expansion and flexibility is accomplished by simply modifying the control program. Chapter IV includes the development of the software program for the Distress Alarm Keyer. Flow charts for the subroutines have been drawn to explain the strategy followed while writing the program.

As presently implemented, the Data collection Platform and the

Distress Alarm Keyer are two separate units which communicate with each other. The microprocessor version of the Distress Alarm Keyer has reduced its physical size to an appreciable extent. Work is also being done on the miniaturization of the Data Collection Platform. The final goal would be to include the Keyer and the DCP in one package. The flexibility of the present design of the Keyer and the prospects of its expansion have been discussed in Chapter V. It also includes a comparison of the hard-wired logic design with the microprocessor-based design of the Keyer.

CHAPTER II

DISTRESS ALARM KEYER

The current Search and Rescue (SAR) system is severely deficient and ineffective. It comprises of Emergency Locating Transmitters (ELT) aboard airplanes and small marine craft. In the event of an emergency a CW beacon is transmitted to alert the SAR forces. Besides having other flaws, the major shortcoming in these transmitters is the low transmitted power (75 mw). Under normal circumstances a search plane would have to be within 20 to 25 miles in order to receive an ELT distress signal. Studies have concluded that low earth-orbiting satellites (600 n. miles) could be used to pick up ELT transmissions of 1 watt. The use of satellite techniques holds tremendous promise for increasing the effectiveness of the SAR program by providing more accurate position data and enhancing the coverage area. The importance of a Satellite-Aided SAR Program has been emphasized in the first section of this chapter. In the event of an emergency a distress message in binary-coded-decimal format can be entered in a Distress Alarm Keyer and then transmitted to the satellite. The message will be relayed to an earth station and the SAR forces will be alerted and deployed. The development and the testing of the hard-wired version of the Keyer has been discussed in the second section. The third section includes the function of the Distress Alarm Keyer. Now a microprocessor version of the Keyer has been developed using Intel's MCS-48 microcomputer family. Its operation is described in the last section of this chapter.

1. Need and Importance of Satellite-Aided SAR System

The U. S. areas of responsibility (as defined in National Search and Rescue Plan) for SAR coverage are widely dispersed over inland, maritime and overseas regions. Distress incident notification and location must be transmitted to the appropriate regional SAR coordinators in a timely fashion. At present, aircraft and "inspected marine vessels" are required by law to carry emergency transmitters. The responsibility for monitoring these transmissions is not clearly defined and at best provides only a sparse coverage of the total geographical area under U. S. responsibility.

Besides aircraft and "inspected marine vessels," small craft present an ever increasing problem in safety assurance in the U. S. coastal waters and offshore. There were over 6 million recreational boats registered in 1975 and roughly 5% of these only have VHF radio or equivalent for distress alerting. Over 1200 deaths occur annually, on the average, within 40 Km of the U. S. coast. There is also a lack of complete communication coverage on the high seas for search and rescue while each year there is an increase in the number of small craft venturing offshore.

The need for a more effective search and rescue system cannot be overstated. NASA has laid the foundation and the cornerstone for a future system through its booster and spacecraft capabilities, and has evolved the basic ranging and communication techniques which are the prime ingredients of a successful SAR system. The number of lives which could be saved annually is by itself sufficient to justify the cost of a search and rescue capability aboard satellites. The obvious advantage

of satellites is to monitor large geographical areas on a regular basis for detection of a distress beacon and also in determining the location of the beacon by techniques such as Doppler tracking. This reduces the time between occurrence and detection of a distress incident. The position location capability of the satellite-aided system gives a distress location with an accuracy of about 5-10 Km thereby reducing the search time appreciably. The reduction in time achieved is directly translatable to fewer casualties and reduced costs of SAR operations.

2. Development of Hard-Wired Distress Alarm Keyer

A hard-wired version of the Distress Alarm Keyer was first developed in 1976. The device consists of a keyboard encoder and a serial-to-parallel converter housed in two separate boxes. Digits entered from the keyboard are converted to binary-coded-decimal format and transmitted serially to the serial-to-parallel converter. The serial-to-parallel converter accepts serial data outputs of the keyboard encoder and presents the last four keyboard entries to the Data Collection Platform on 16 parallel lines.

The Distress Alarm Keyer was used in an experiment conducted aboard a 33-foot shoal draft sloop while cruising in the area of the Bahamas from Dec. 18, 1976 to Jan. 14, 1977 with the low altitude NIMBUS-6 Satellite. The purpose of this satellite communication and position location experiment was to evaluate the reliability of using satellites with low cost, low power mobile equipment for search and rescue application. The results of the experiment demonstrated the feasibility and practicality of using a low-orbit satellite system as an aid to Search and Rescue operations. It was observed that the accuracy of Doppler-

determined altitude and longitude of a transmitting source would reduce the radius of search to 2 miles most of the time. The experiment also demonstrated a high degree of reliability of transmitting messages (by simple code) using a low power, low cost keyboard encoding device.

During the experiment it was observed that entering the data sequences into the Keyer was an inconvenient and laborious process. The operator had to enter one four-digit piece of data every minute during the satellite overpass. The uncertainty of the exact pass time caused the operator to begin entering the sequence early and repeat it 3 or 4 times. This procedure required approximately 40 minutes.

A modification to this effect was carried out in the Distress Alarm Keyer design in 1977. Design of the serial-to-parallel converter was modified so that up to nine messages (four digits each) could be stored in expanded shift registers which would automatically sequence and recycle the messages to the output.

In order to enter the messages in the modified Distress Alarm Keyer, the following procedure is used. First, a multi-position switch located on the top of the serial-to-parallel converter box is set to the desired number of messages to be input. One, two, four, six, eight, or nine messages are allowed. Next, the set number of four-digit messages is keyed in. The serial-to-parallel converter will now automatically handle the data.

The first message input will be located in the output register and the following messages will be stored in an additional shift register. At the appropriate time, a carrier-on signal will come from the Data Collection Platform. This signal will cause the data to shift such that

the second message input will go to the output register, and the first message input will shift into the register previously filled with the last message input. Therefore, repeated carrier-on signals (which will occur at appropriate times) will effectively shift the data around in a loop, each message passing through the output register as many times as needed during the satellite pass. The new data can only be keyed in when previous data is not being shifted.

3. Function of the Distress Alarm Keyer

In the case of an emergency, the user can enter the distress message, identification data, weather conditions, estimated position etc. in the Distress Alarm Keyer from a keyboard. As presently implemented, a total of nine messages can be entered into the Keyer. Each message, consisting of one to four digits, is stored in the memory in binary-coded-decimal format. Each digit of the one to four digit coded message uses four of the (typically) 32 parallel input bits provided by the standard format of the DCP. The number of distinct messages can be increased to 10,000 by connecting all four of the digit outputs to two of the DCP parallel sensor inputs. When the DCP is not required to transmit data from any other device, the four message digits can also be connected to the remaining two parallel sensor inputs so that the message will be sent twice during each transmission. This technique improves the performance of the system under the influence of short noise bursts. The user must know whether the Distress Alarm Keyer has been wired to transmit one, two, three or four digits and should enter the messages accordingly.

4. Operation of the Distress Alarm Keyer

After depressing the Reset key, the user must store the digit corresponding to the total number of messages he desires to enter in the Distress Alarm Keyer. This is done by depressing the "NUMBER" key and then entering that digit. A maximum of 9 messages can be stored in the Keyer. The LED's will display "┌┐.T" (T = the digit corresponding to the total number of messages).

In order to enter a message in the Distress Alarm Keyer, the user depresses the "ENTER" key and the key corresponding to the position in which he desires to store the message. Now he can enter the four digits of the message. The LED's will display "EY - XXXX" (Y = the position of message, X = the code of message).

The user can read the stored message by depressing the "READ" key and the key corresponding to the position of the message he desires to read. The LED's will display "┌ Y - XXXX" (Y = the position of message, X = the code of message).

Each time a positive pulse is received from the DCP or the "OUTPUT" key is depressed, a message is transmitted to the DCP. The messages are transmitted sequentially from the first message to the last message. After transmitting the last message the next message to be transmitted will be the first one. Each time a message is transmitted, the LED's will display "├ Y - XXXX" (Y = the position of message, X = the code of message).

CHAPTER III

HARDWARE DEVELOPMENT OF THE DISTRESS ALARM KEYER

The microprocessor version of the Distress Alarm Keyer has been developed around Intel's MCS-48 microcomputer family. The Keyer was realized using the 8035 microprocessor. The program was stored in an external program memory, the output of which was connected to the data BUS of the 8035. The Intel 8279 Programmable Keyboard/Display Interface chip was used to interface the keyboard and the LED display. The circuit has been discussed in detail in this chapter.

In the last section it has been proposed to use the 8748 instead of the 8035 microprocessor. The 8748 and the 8035 processors are pin-to-pin compatible and execute the same instruction set. The only difference between the two is that the 8748 has an internal 1K bytes x 8 bits PROM which is used to store the program. This eliminates the need of an external program memory and the associated circuitry and makes the circuit much simpler.

1. Hardware Description Using 8035 Microprocessor

The 8035 is an 8-bit microprocessor and contains 64x8 RAM data memory, 27 I/O lines, and an 8-bit timer/counter in addition to an on-board oscillator and clock circuits. The detailed description of the 8035 has been provided in Appendix A. The circuit for the Distress Alarm Keyer is shown in Figure III.1. A 4x4 keyboard has been used to generate various commands and to enter messages in the data memory. The commands and the messages are displayed on seven 8-segment LEDs.

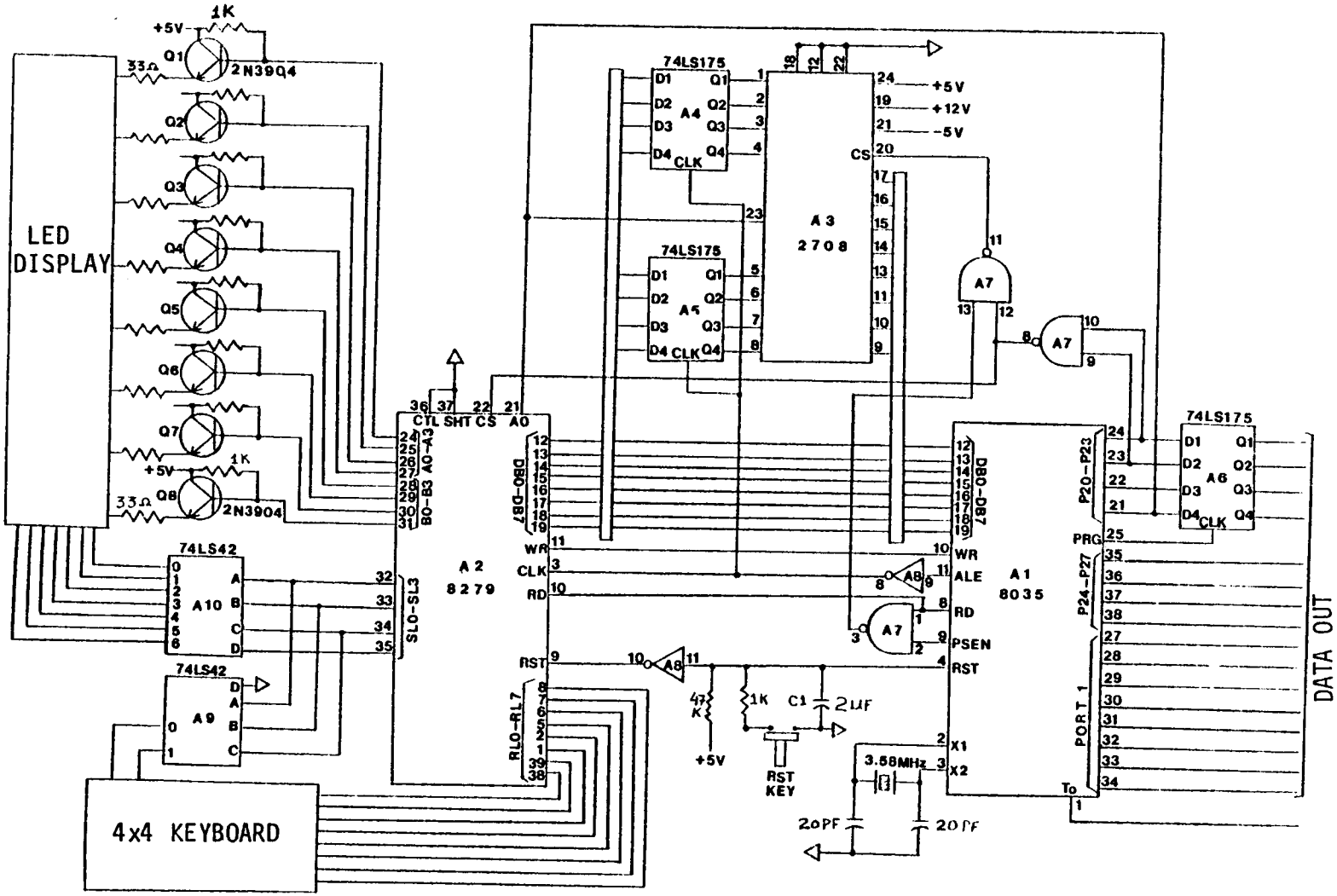

Figure III.1 Distress Alarm Keyer (using 8035 microprocessor)

The Data Bus of the processor is shared by the external program memory and the keyboard/display interface chips. The detailed description of the circuit is discussed in the following sections.

1.1 Clock

The 8035 has an on-board oscillator which is a high-gain series-resonant circuit with a frequency range of 1 to 6 MHz. A 3.58 MHz crystal connected between X1 and X2 pins provides the feedback and phase shift for the oscillation. This generates a machine cycle time of 4.2 μsec (3.58 MHz ÷ 3 ÷ 5). The clock signal is provided continuously on Address Latch Enable output pin.

1.2 Reset

The reset input provides a means for initialization for the circuit. The 8035 has an internal pull-up resistor which in combination with C1 (2 μF) provides an internal reset pulse of sufficient length to guarantee all circuitry is reset during the power-up mode. C1 is also connected to the inverter A8-11 that provides the reset input for the 8279 (refer to Appendix B for the description of the 8279 chip). An external reset pulse can also be generated by depressing the reset key for at least 50 millisecond.

1.3 Chip-Select Logic

The 8035 has an 8-bit bidirectional BUS (DB0-DB7). The BUS port has been used to communicate with the 8279. It is also being shared with the external program memory chip A3 (2708 EPROM, 1kx8 Byte) to fetch the program data. The lower half of Port 2 (P20-P23) selects

either of the chips under the program control. Whenever the 8279 is to be addressed, 1-1 is written to P23-P24. These pins are connected to NAND gate A7, the output of which selects the 8279. P20 is connected to pin 21, Buffer Address, of the 8279. A high on this line indicates the signals in or out are interpreted as a command or status. A low indicates that they are data.

The output of NAND gate A7-11 selects the 2708 during Fetch/Read cycles provided the 8279 was not being addressed i.e. output A7-8 was high.

1.4 Program Fetch

Address Latch Enable (ALE) signal occurs during each machine cycle. The address is valid on the lower half of Port 2 and on the data BUS at the negative edge of ALE. The low order 8 bits of address present on the BUS are latched into A4 and A5 (74LS 175) at the positive edge of $\overline{\text{ALE}}$. The address bits present on Port 2 do not require an external latch since the address remains unchanged on Port 2 during the complete fetch cycle. The contents of program memory are then loaded into the 8035 at next positive edge of ALE.

ALE signal has also been used as a clock input to the 8279.

1.5 Output

Ports 1 and 2 of the 8035 have been used to transmit 16 bits of data to the Data Collection Platform. These ports are each 8 bits wide and have identical characteristics. Data written to these ports is statically latched and remains unchanged until rewritten. The lower half of Port 2 has been used in three different modes. Firstly, it has

been used as an output port. Secondly, it provides high-order address bits for external program memory. Thirdly, it controls the chip select lines of A2 (8279) and A3 (2708). The output data should remain unchanged during the transmission of each message to the Data Collection Platform. But the output of lower half of Port 2 is changed under program control to perform various functions. In order to use it as an output port for data transmission, an external latch A6 (74LS175) has been used. The output present on the lower half of Port 2 is latched into A6 on the positive edge of PROG strobe.

1.6 Keyboard

A 4x4 keyboard has been employed to generate various command signals and to enter data into the 8035. The position of different keys is shown in Figure III.2. The keyboard has been wired to form a 2x8 matrix which is scanned by two scan-row lines. These lines have been generated by decoding the scan lines SL0-SL2 of the 8279 by A9, 74LS42 3-to-8 Decoder. Wiring of the keyboard is shown in Figure III.3. When a key is depressed, the corresponding return line is pulled low. The change is detected by the 8279 and its debounce logic is set. A full scan of the keyboard is ignored, then other depressed keys are looked for. If none are encountered, it is a single key depression and the key position is entered into the FIFO.

The 8279 has been used in the scanned-keyboard, 2-key lockout mode. In this mode a key depression generates a 6-bit encoding of the key position which is entered into FIFO. The least significant three bits are from the column counter and indicate to which return line the key

NUMBER	ENTER	READ	RST
Tx	8	9	x
4	5	6	7
0	1	2	3

Figure III·2 Position of Keys

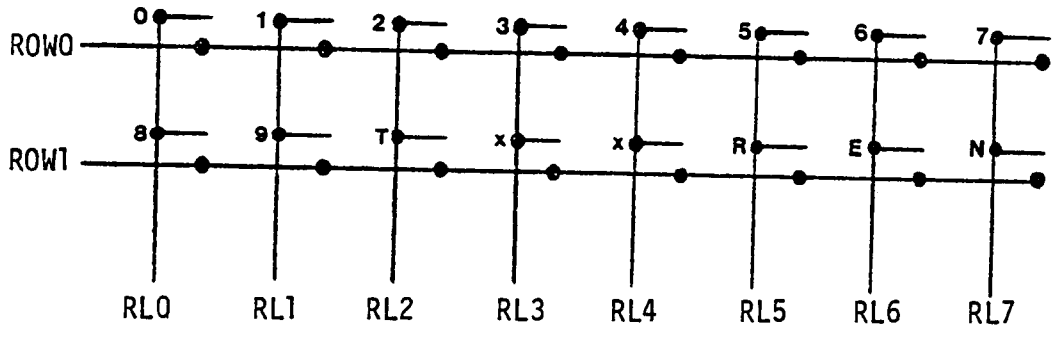

Figure III·3 Keyboard Wiring

was connected. The next three bits are from the scan counter and indicate the row the key was found. The data format is shown in Figure III.4.

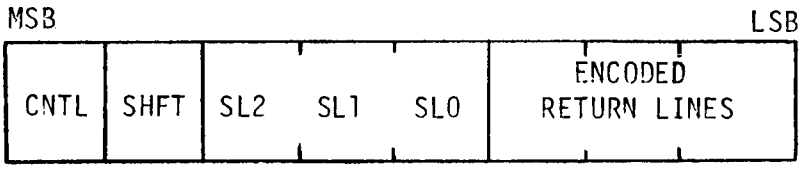

Figure III·4 Scanned Keyboard Data Format

1.7 Display

The 8279 has been used in 8-bit character, left entry display format. In this format each display position directly corresponds to a byte in the Display RAM. Address zero in the RAM is the left-most display character and address 15 is the right-most display character. Entering characters from position zero causes the display to fill from the left.

Seven 8-segment common-cathode LEDs have been used for the display. Transistors Q1-Q8 serve as buffer between the LEDs and the eight display output lines (A0-A3, B0-B3) and drive the LEDs. The data from these output lines is synchronised to the scan lines (SL0-SL3) for multiplexed digit displays. The scan lines have been decoded by A10, 74LS42 3-to-8 decoder, and its outputs are connected to the common cathodes of the LEDs.

1.8 Single Step

This feature provides a debug capability in that processor can be stepped through the program one instruction at a time. Whenever a low level is applied on $\overline{SS}$ input, the processor responds by stopping during the instruction fetch portion of the next instruction and ALE is raised high. In this state the address of the next instruction to be fetched is present on BUS and the lower half of Port 2. $\overline{SS}$ is then raised high to bring the processor out of the stopped mode allowing it to fetch the next instruction. The exit from the stop is indicated by the processor bringing ALE low. To stop the processor at the next instruction $\overline{SS}$ must be brought low again as soon as ALE goes low. A timing diagram showning interaction between ALE and SS is shown in Figure III.5.

The single-step function has been implemented by using a D-type flip-flop, 7474. The circuit is shown in Figure III.6. Whenever ALE goes low it brings $\overline{SS}$ low via the clear input. The processor is now in stopped state. The next instruction is initiated by clocking a "1" into the flip-flop. This "1" will not appear on $\overline{SS}$ unless ALE is high removing clear from the flip-flop. In response to $\overline{SS}$ going high the processor brings an instruction fetch which brings ALE low resetting $\overline{SS}$ through the clear input and causing the processor to again enter the stopped state.

2. Hardware Description Using 8748 Microprocessor

The proposed circuit of the Distress Alarm Keyer using the 8748 microprocessor is shown in Figure III.7. It is very similar to the circuit using the 8035 microprocessor.

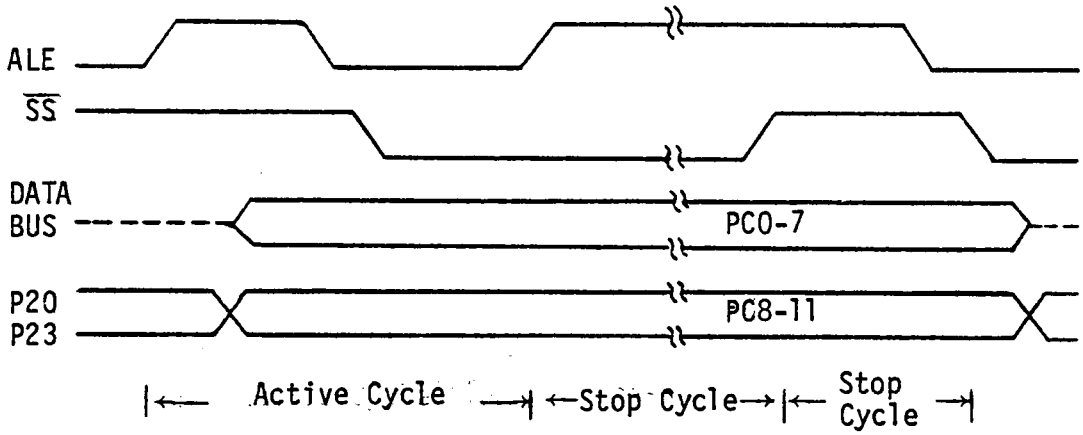

Figure III·5 Single Step Timing

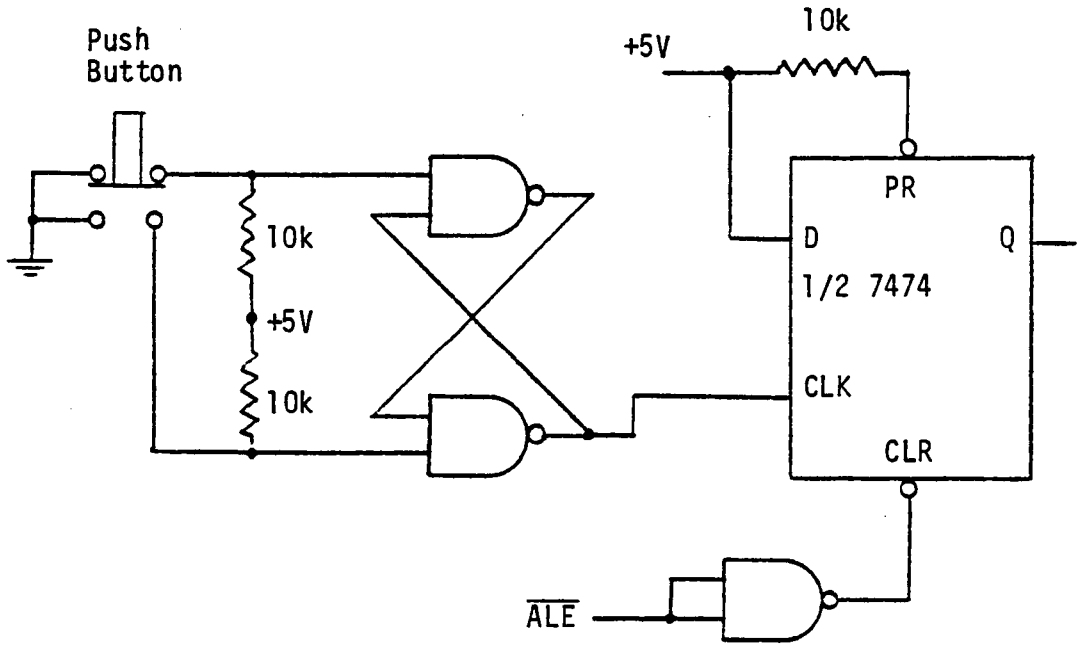

Figure III·6 Single Step Circuit

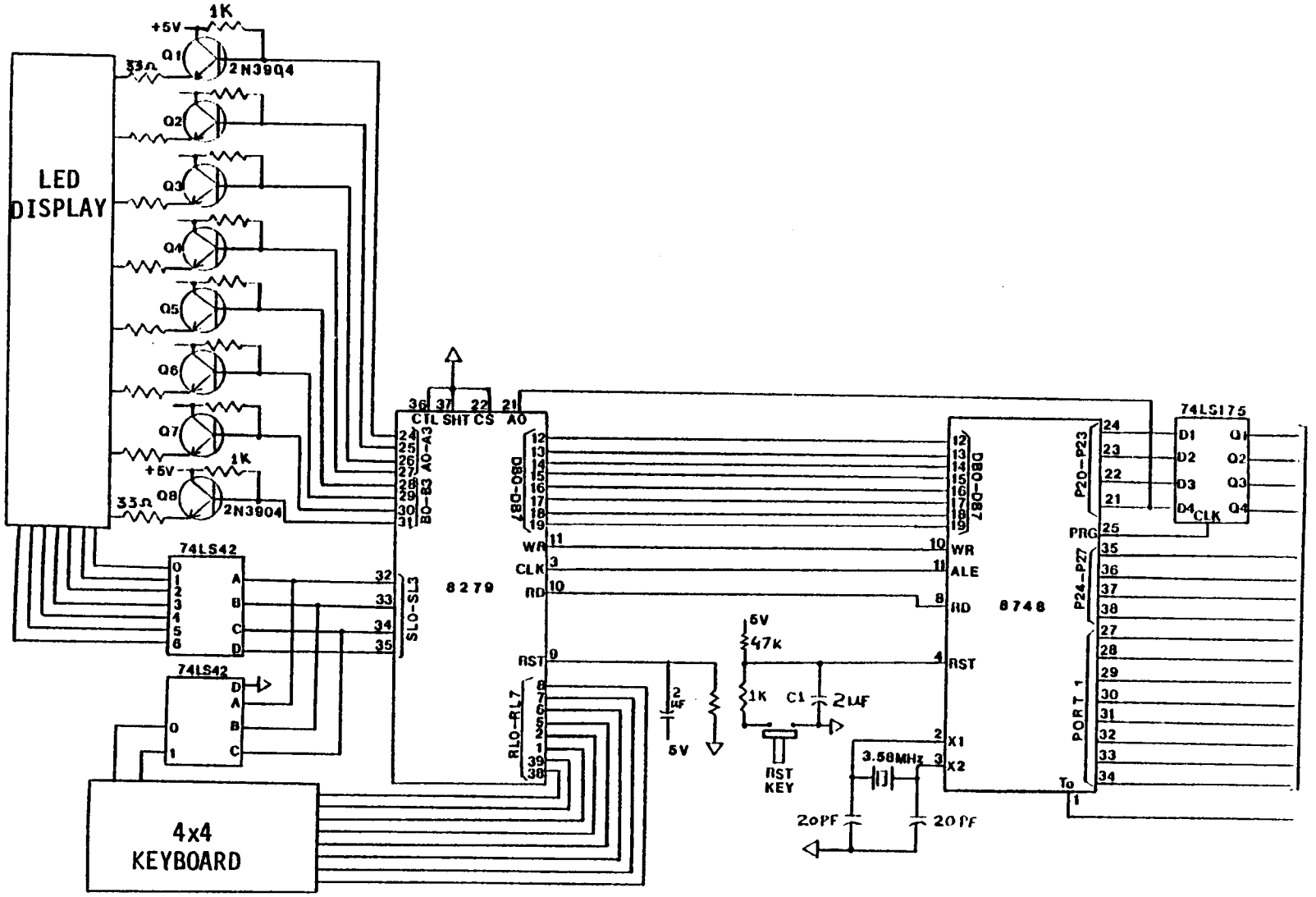

Figure III·7 Distress Alarm Keyer (using 8748 microprocessor)

The program can be stored in the internal program memory of the 8748. This eliminates the need of an external PROM and also its associated address latches. The data BUS of the processor is now exclusively used to communicate with the keyboard interface chip, the 8279. Therefore the chip-select logic is no more required. The CS pin of the 8279 is permanently tied to ground. Besides these changes in the circuit, the rest of it is exactly similar to the circuit using the 8035, which has already been discussed earlier in this chapter.

CHAPTER IV

SOFTWARE DEVELOPMENT OF THE DISTRESS ALARM KEYER

The control sequence in a microprocessor system is stored and executed as a software program. Expansion and flexibility is accomplished by simply modifying the control program. In contrast, modifications of a hardwired logic design would require redesigning and laying out the new circuit diagram, utilizing additional components, and rewiring the layout. The development of a software program for the Distress Alarm Keyer has been discussed in this chapter. The program has been divided into a number of subroutines. Each subroutine corresponds to a specific command and performs a dedicated task. Any command received from the keyboard is decoded by the program and the corresponding subroutine is addressed to perform the required task. Nine distress messages (4 BCD digits each) are stored in the Data Memory (in locations 34 to 51) of the 8035. Each time the message data is keyed in, its validity is checked. It is only stored in the memory if its value is not greater than decimal nine.

During the software development of the Distress Alarm Keyer Register Bank 0 has only been used. Therefore, Register Bank 1 and the data memory locations 52 to 63 can be utilized during the future expansion phase of the Distress Alarm Keyer.

The description of the subroutines is included in the following sections. Each subroutine has been explained with the help of flow charts. At the end of this chapter the code of the program for the Distress Alarm Keyer has also been included.

1. Reset Routine

Activating the reset line of the microprocessor by applying power to the Distress Alarm Keyer or by depressing the reset key, causes the first instruction to be fetched from location zero. This is the starting location of the reset routine. Firstly, the output ports and the data memory locations, which store the distress messages, are cleared. This is done to ensure no erroneous message is transmitted to the Data Collection Platform. Secondly, the LED display is blanked and the clock frequency of the 8279 is preset. Now the program waits for the operator's command. If a valid command is received, it is decoded and the program jumps to the corresponding subroutine. The flow chart of the "RESET" routine is drawn in Figure IV.1.

2. Subroutine "Number"

This subroutine accepts the number of total messages to be transmitted. In the beginning of the subroutine, the Keyer acknowledges the operator's command by displaying "⌐." The program waits for any key depression. If a valid key is depressed (i.e. number of total messages does not exceed 9), it is stored in R7 and also displayed on the LEDs. On the contrary, an invalid key depression is not acknowledged. The display is blanked and the program jumps back to the reset routine. The flow chart of the "Number" subroutine is drawn in Figure IV.2.

3. Subroutine "Enter"

This subroutine receives the message data from the keyboard and enters it in the data memory. Message number 1 is stored in locations 34 (higher byte) and 43 (lower byte), message number 2 is stored in

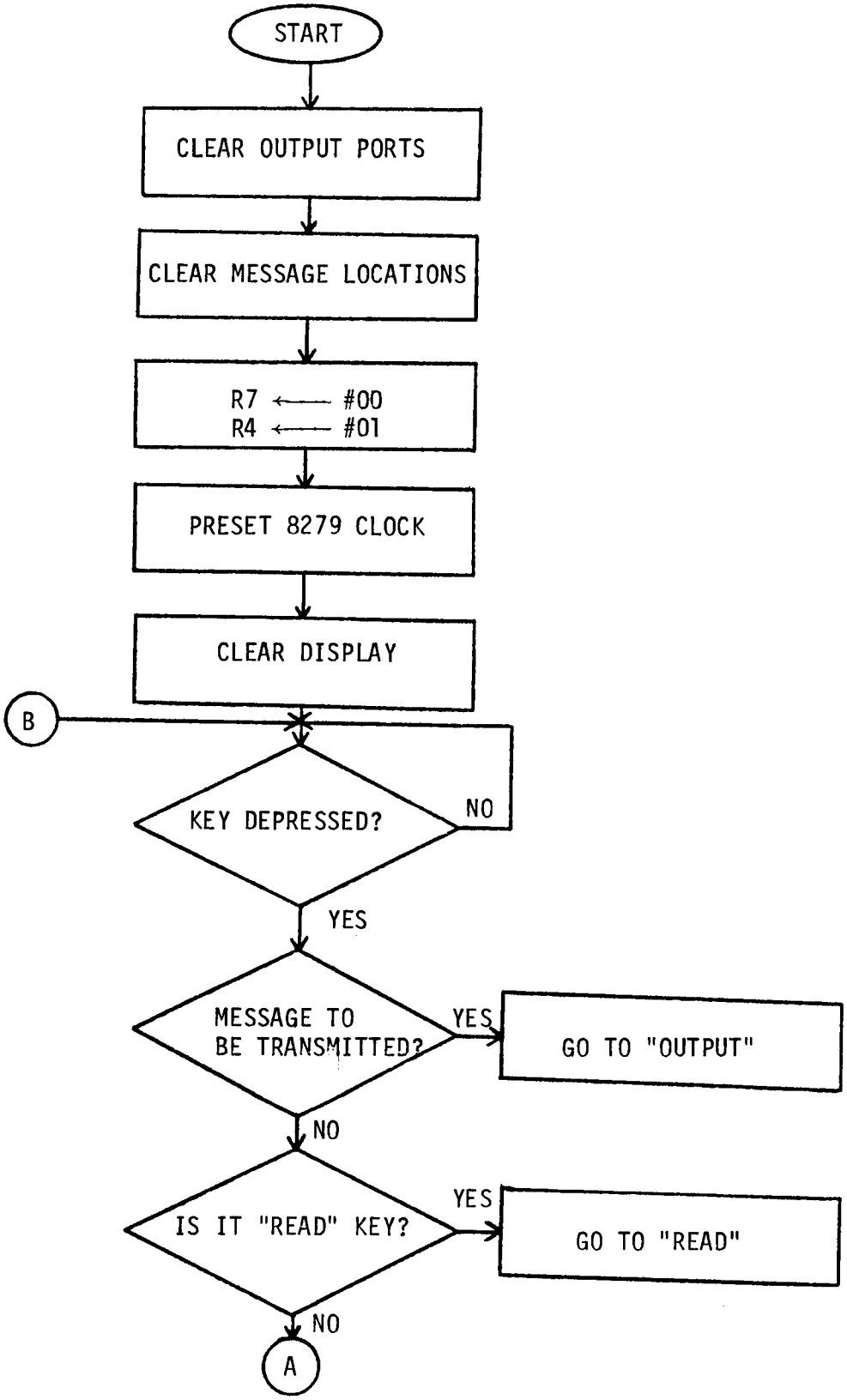

Figure IV·1 "Reset" Routine

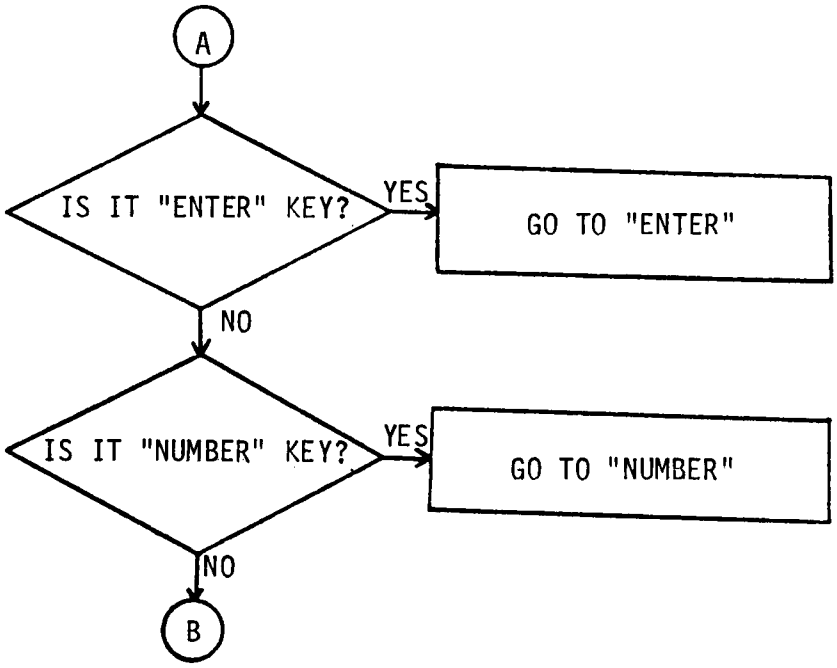

Figure IV.1 (continued)

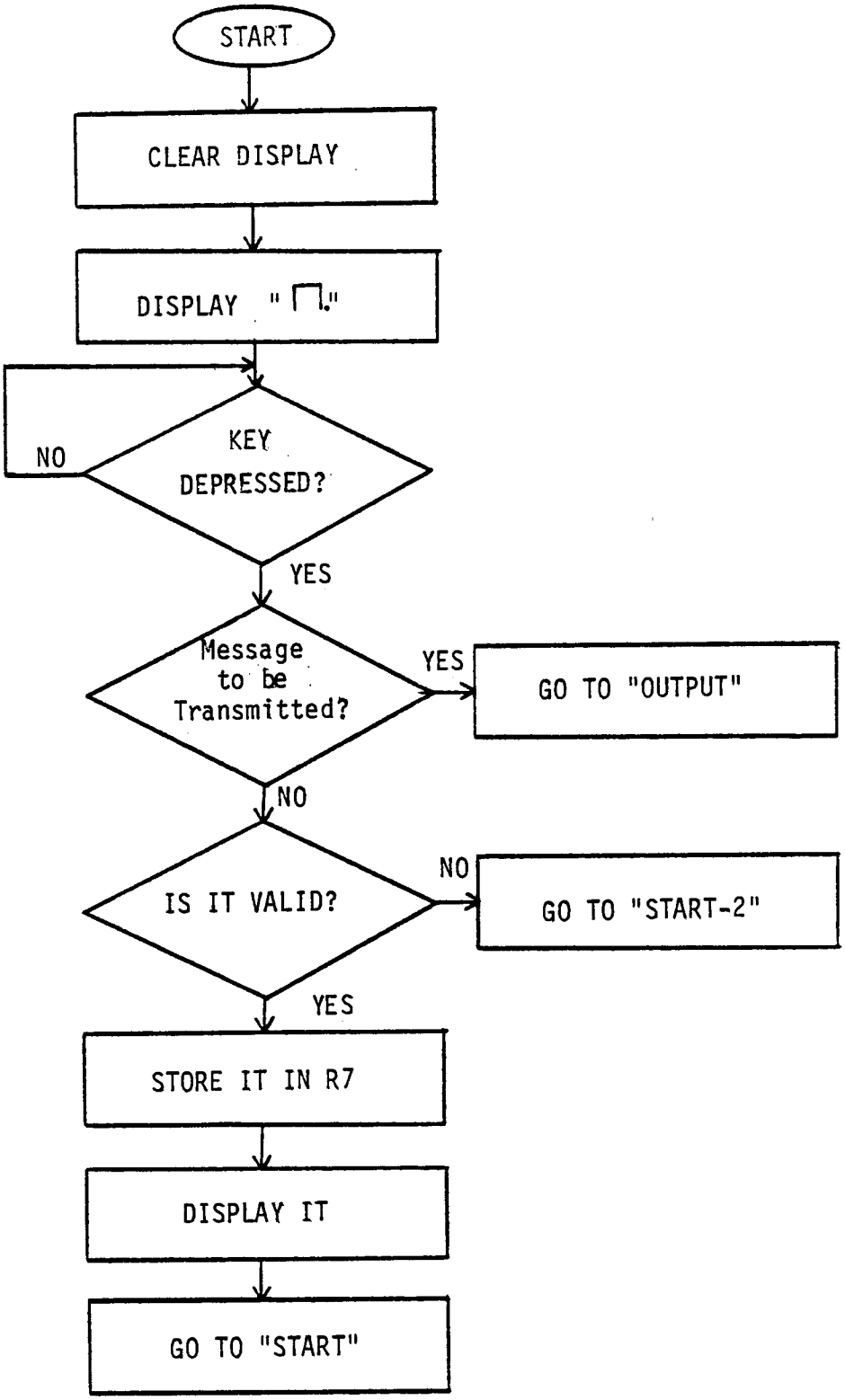

Figure IV. 2. Subroutine "NUMBER"

in locations 35 and 44, and so on. In the beginning of the subroutine, operator's command is acknowledged by displaying "E." The next depressed key is treated as the message number. If it is a valid one, it is displayed on the LEDs and temporarily stored in R2. The keyer then displays " - ". The storage address of the higher byte of the message is calculated and saved in R0. R5 is cleared which indicates that the next two key depressions will contain the higher byte data of the message. These two digits are displayed and stored in the data memory location pointed to by R0. Similarly, the storage address of the lower byte of the message is calculated and saved in R0. R5 is incremented indicating that the next two key depressions will contain lower byte data of the message. These two digits are also displayed and stored in the data memory. The flow chart of the "ENTER" subroutine is drawn in Figure IV.3.

4. Subroutine "Read"

This subroutine displays the contents of a message already stored in the data memory. The operator's "READ" command is acknowledged by displaying "⌐." The program waits for the next key depression which contains the number of the message to be displayed. This number and a " - " is displayed on the LEDs. Address of the message is calculated and stored in R0. The contents of the data memory location pointed to by R0 are displayed. This is the higher byte of the message. Address of the lower byte is calculated next. Similarly the lower byte of the message is displayed. The flow chart of "Read" subroutine is drawn in Figure IV.4.

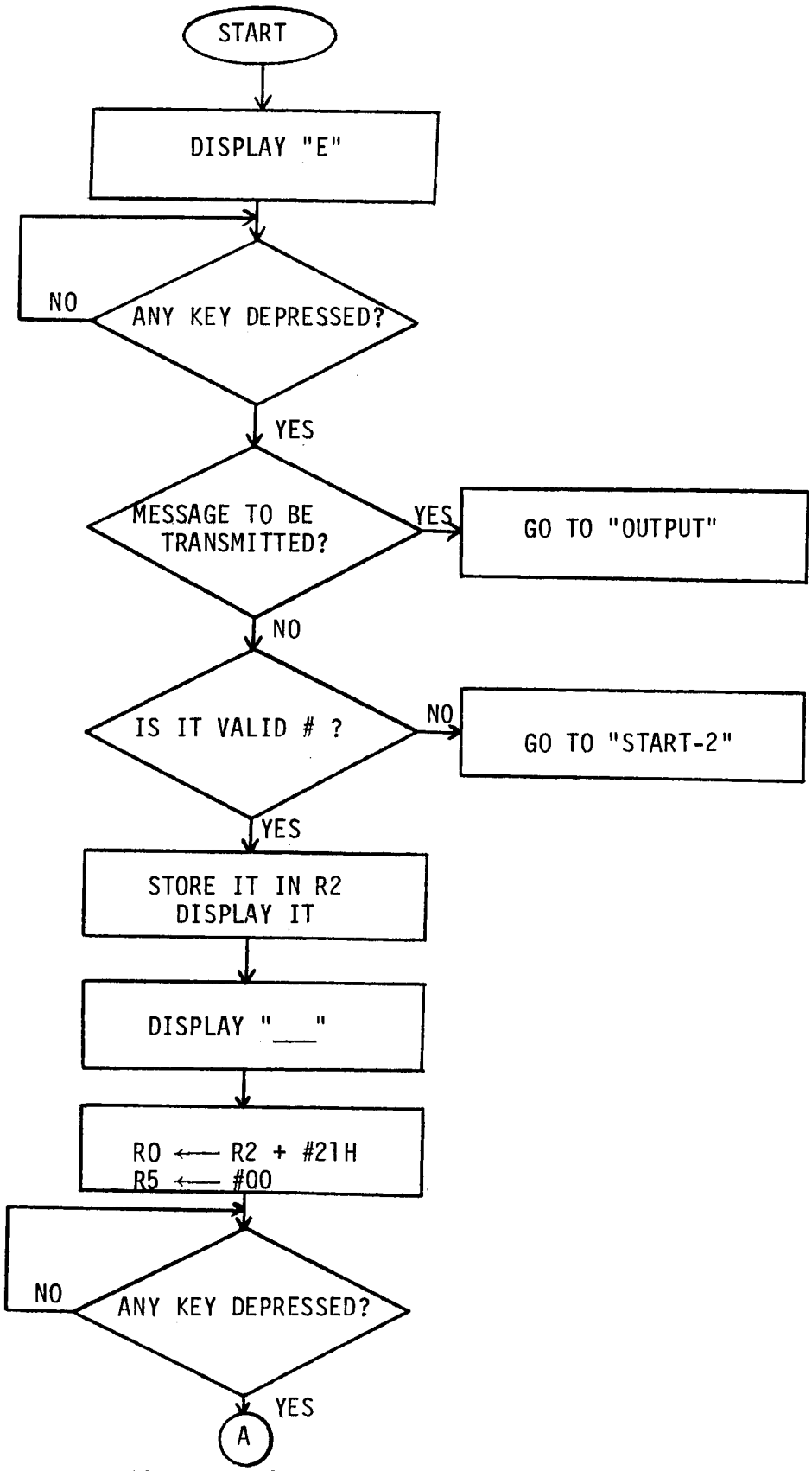

Figure IV·3 Subroutine "ENTER"

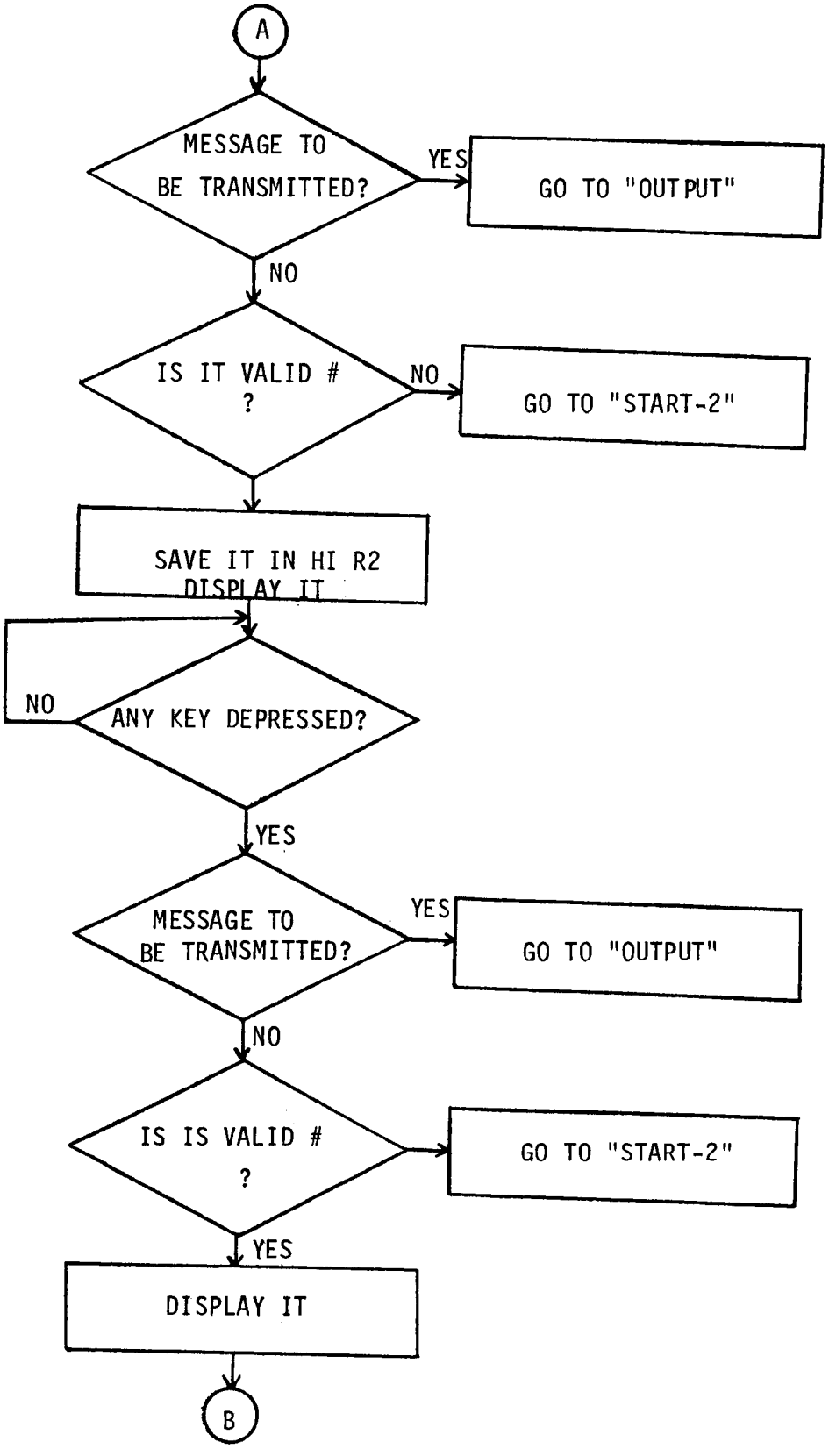

Figure IV·3 (continued)

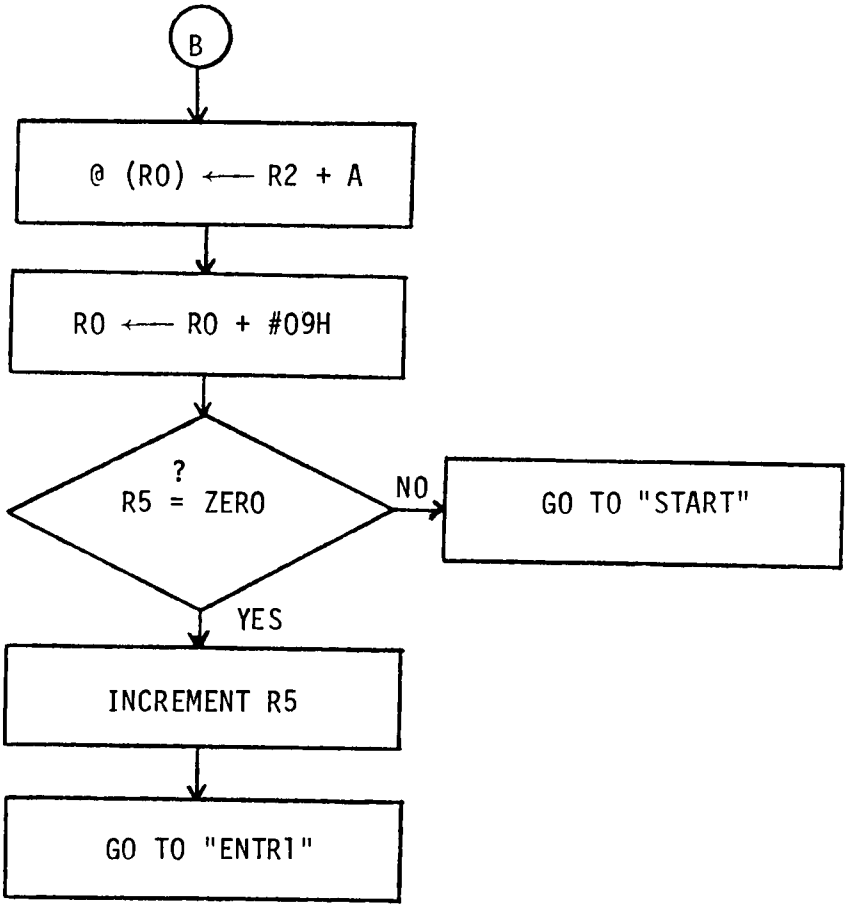

Figure IV.3 (continued)

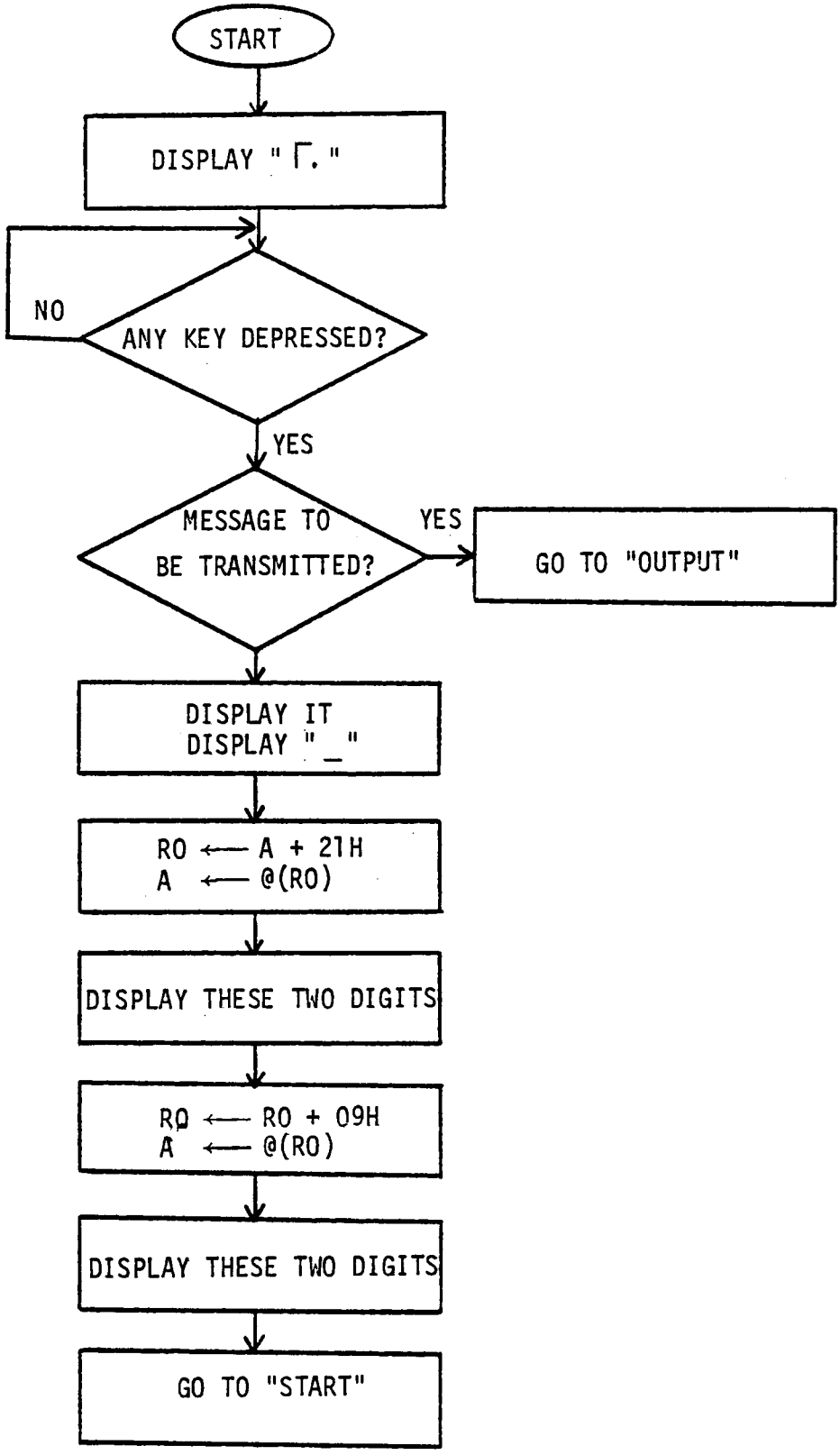

Figure IV.4 Subroutine "READ"

5. Subroutine "Output"

When an output command is received by the Keyer, this subroutine latches the contents of the next serial message to the output ports and also displays it on the LEDs. The registers used in this subroutine are R0, R4 and R7 of Register Bank 0. R0 has been used as temporary storage register. The contents of R4 point to the next message to be transmitted. The number of total messages is contained in R7. In the beginning of the subroutine, the contents of R7 are examined. If its contents are zero then no message is transmitted. On the contrary, the output command is acknowledged by displaying "├ ". The Keyer then displays the contents of R4 and a " - ". Address of the higher byte of the message is calculated and stored in R0. The contents of data memory location pointed to by R0 are latched on the Port 1 of the microprocessor and also displayed on the LEDs. Similarly, the lower byte is latched on the Port 2 and displayed on the LEDs. The contents of R4 and R7 are compared. If they are equal, it indicates that the message transmitted was the last in the sequence. The next message to be transmitted would be the first, therefore, "1" is stored in R4. If the contents of R4 and R7 are not equal, then R4 is simply incremented. The flow chart of the "Output" subroutine is drawn in Figure IV·5.

6. Subroutine "Program 8279" (PG8279)

This subroutine transmits the data stored in the accumulator to the control port of the 8279 chip. No register is used in this subroutine. The flow chart of the "PG8279" subroutine is drawn in Figure IV·6.

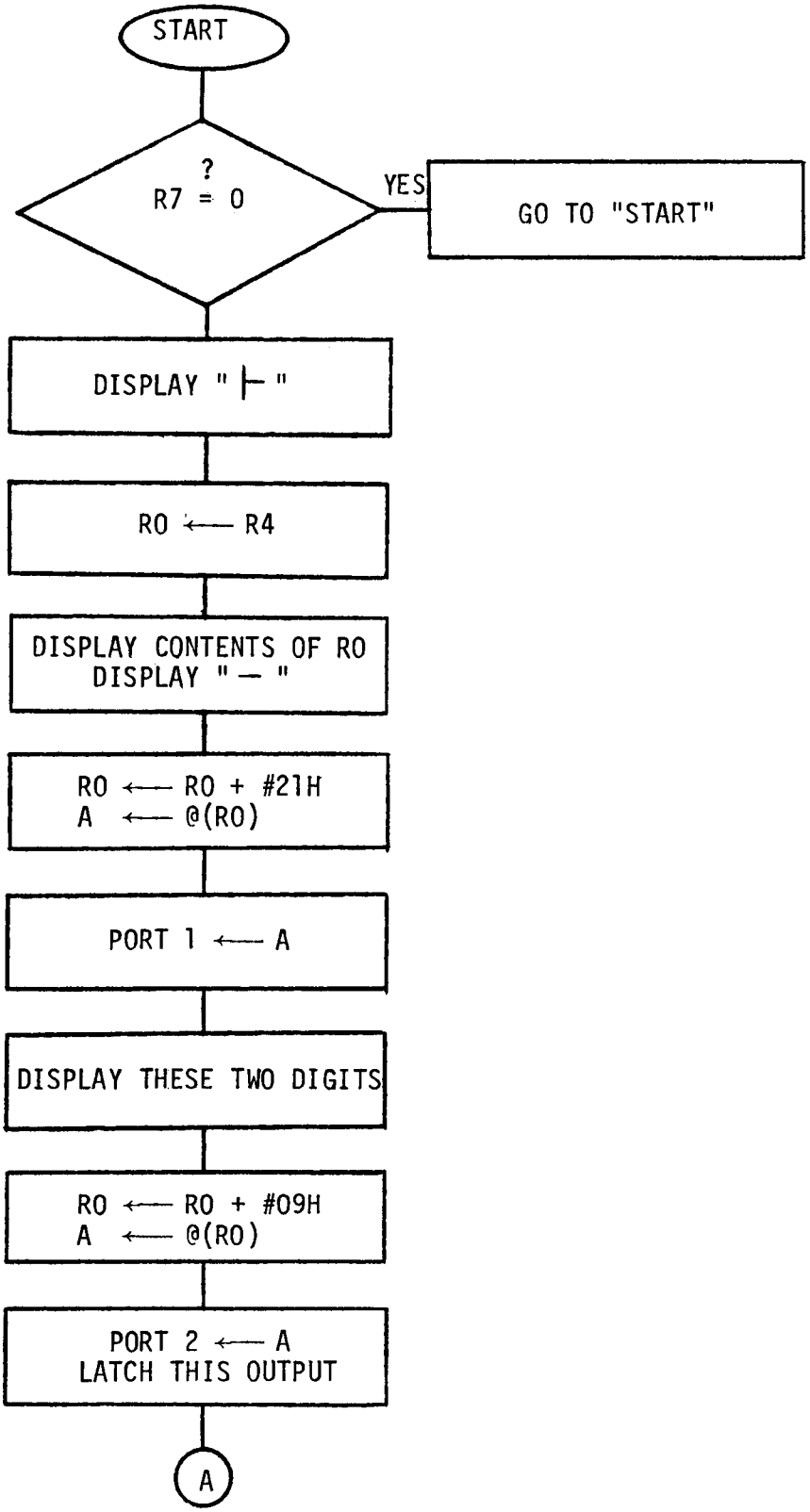

Figure IV·5 Subroutine "OUTPUT"

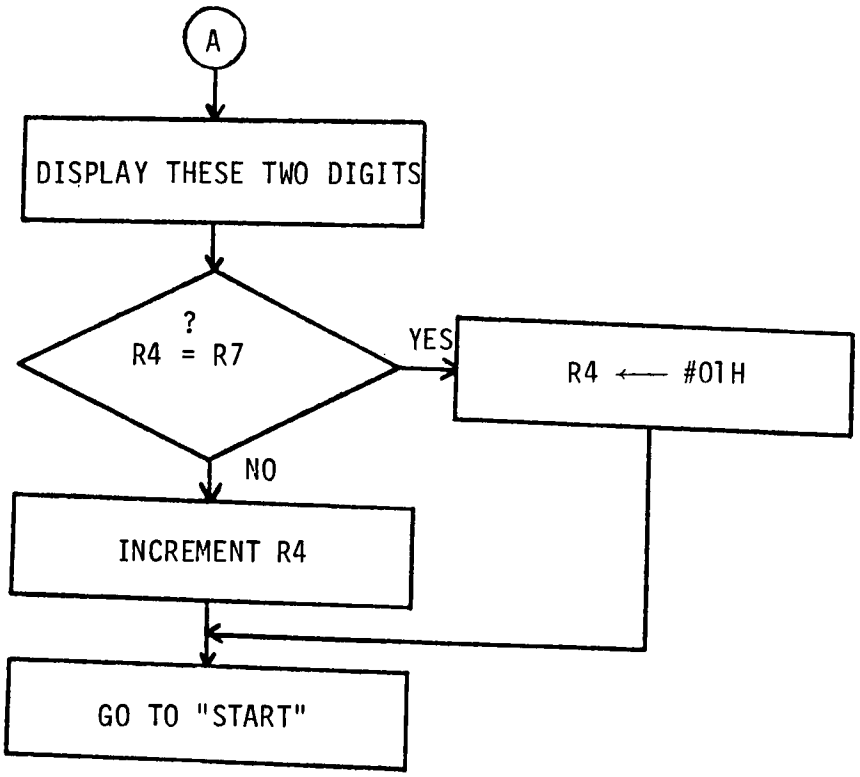

Figure IV. 5 (continued)

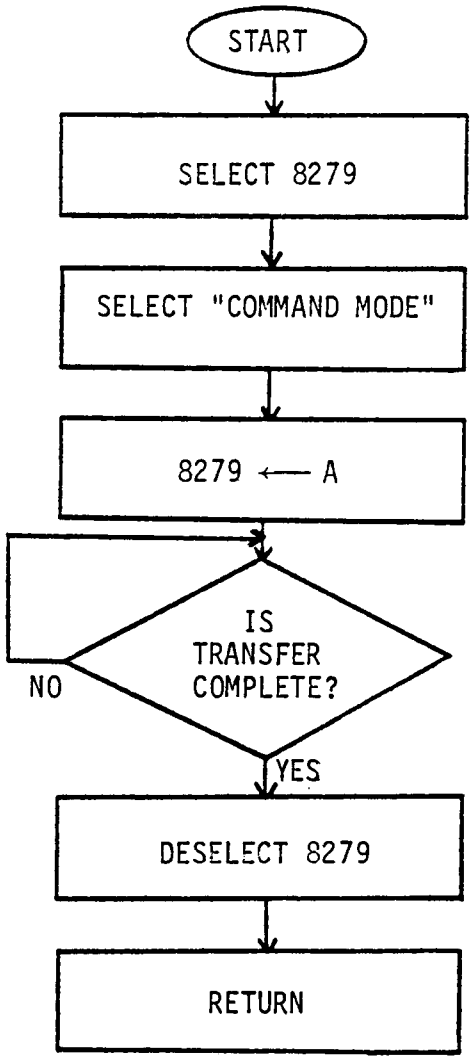

Figure IV. 6 Subroutine "PROGRAM 8279"

7. Subroutine "Blank"

This subroutine blanks the display and resets the next display position to zero. No register is used in this subroutine. The flow chart of the "Blank" subroutine is drawn in Figure IV.7.

8. Subroutine "Display"

This subroutine displays the lower byte of the accumulator to the next display position. No register is used in this subroutine. The flow chart of the "Display" subroutine is drawn in Figure IV.8.

9. Subroutine "Display Number" (DISPNR)

This subroutine displays the BCD digit in the lower byte of the accumulator to the next display position. No register is used in this subroutine. The flow chart of the "DISPNR" surboutine is drawn in Figure IV.9.

10. Subroutine "Display Register" (DSPREG)

This subroutine displays the BCD digits in the higher and the lower bytes of the accumulator at the next 2 characters of the display. R6 of Register Bank 0 is used in this subroutine. The flow chart of the "DSPREG" is drawn in Figure IV.10.

11. Subroutine "Keyboard Input" (KBIN)

This subroutine reads any depressed key and stores its value in the lower byte of the accumulator. At the same time the flag 0 is also monitored. If the flag is set then decimal 10 is stored in the accumulator. No register is used in this subroutine. The flow chart of the "KBIN" is drawn in Figure IV.11.

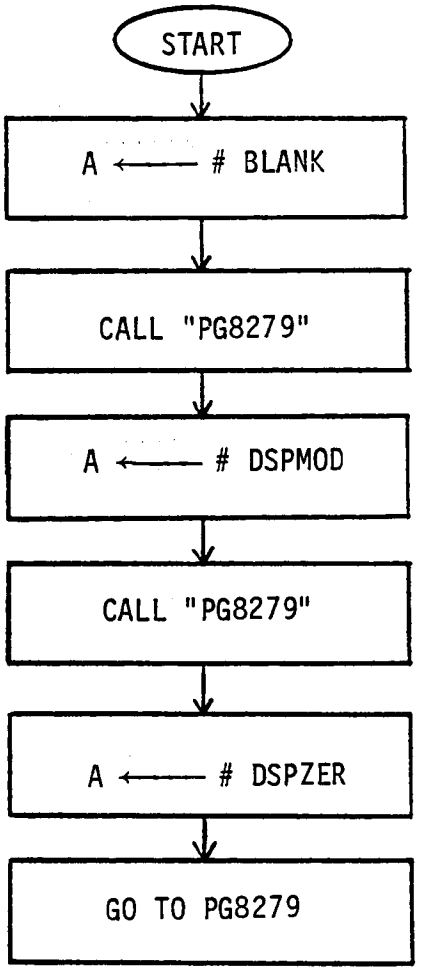

Figure IV. 7. Subroutine "BLANK"

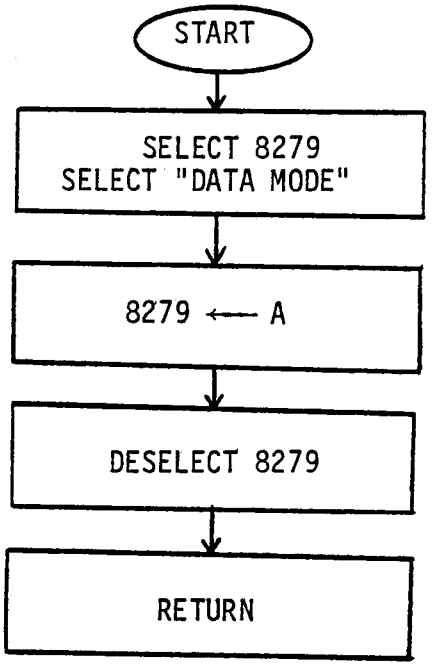

Figure IV. 8 Subroutine "DISPLAY"

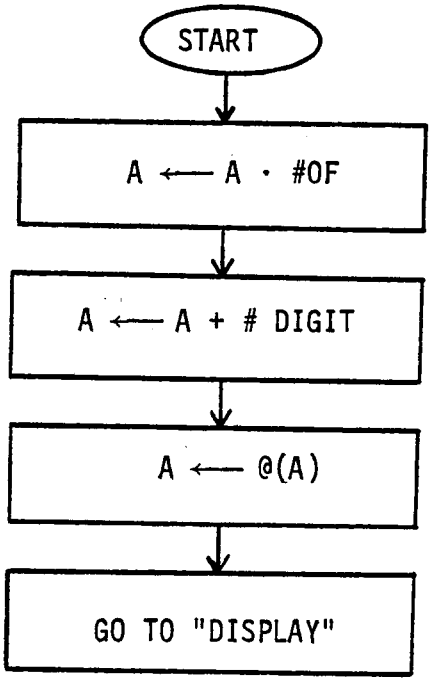

Figure IV. 9 Subroutine "DISPLAY NUMBER"

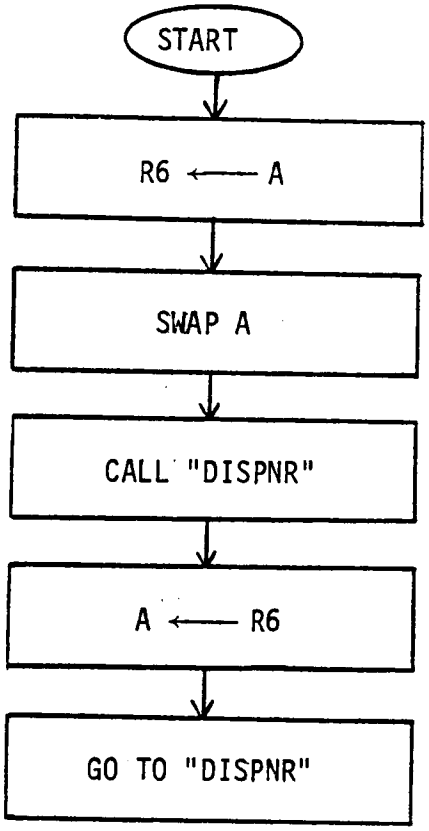

Figure IV. 10. Subroutine "DISPLAY REGISTER"

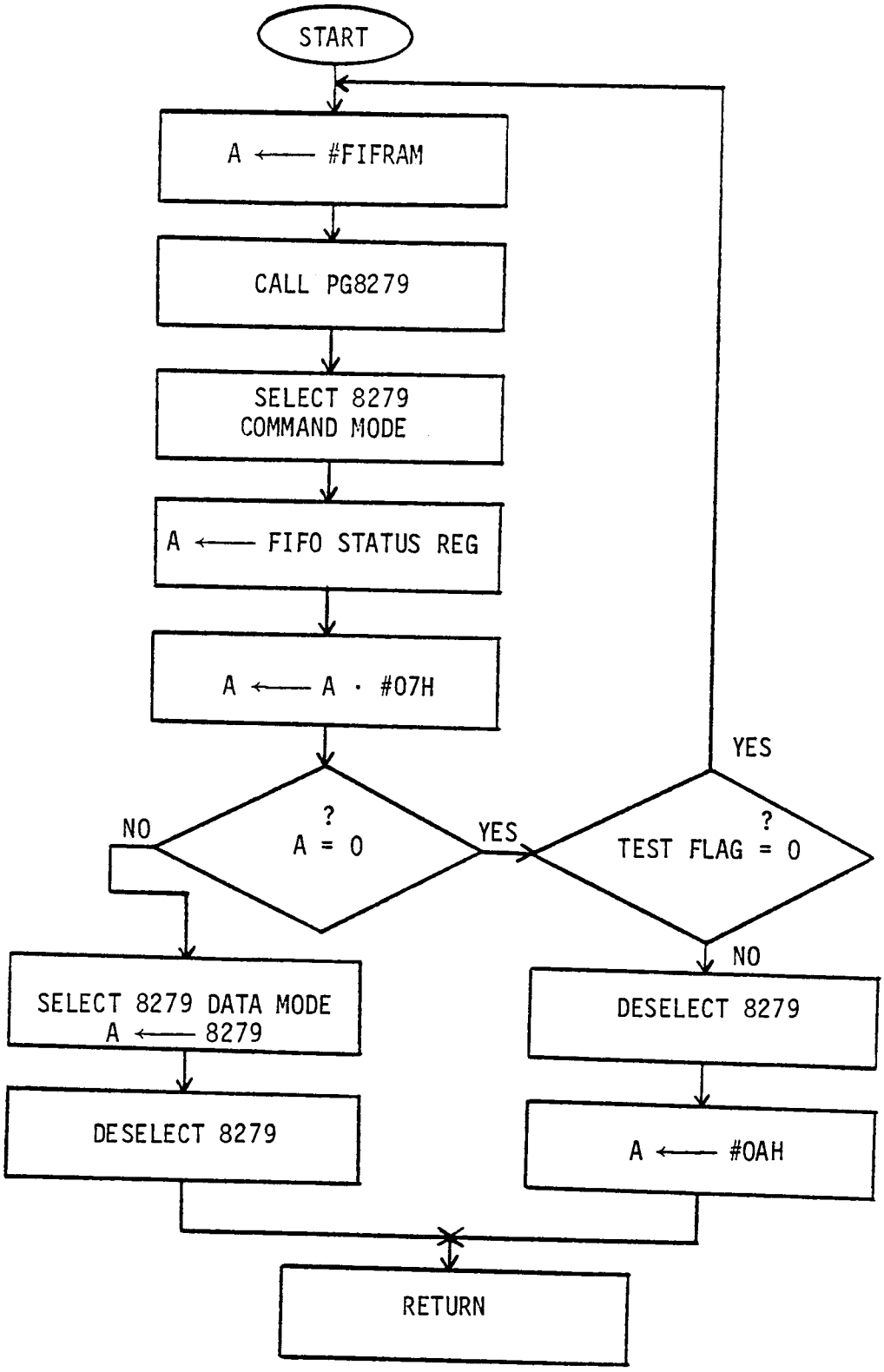

Figure IV. 11. Subroutine "KEYBOARD INPUT"

12. Program Code of the Distress Alarm Keyer

```
                    "SYSTEM EQUATES"

                    KEYPAD INPUTS

Key 0       =       0
Key 1       =       1
Key 2       =       2
Key 3       =       3
Key 4       =       4
Key 5       =       5
Key 6       =       6
Key 7       =       7
Key 8       =       8
Key 9       =       9
Transmit    =       A       ; Transmit Next Message to DCP
Not Used    =       B
Not Used    =       C
Read        =       D       ; Display the Message
Enter       =       E       ; Enter the Message
Number      =       F       ; Number of Total Messages
```

DISPLAY SEGMENT CODES

```
Digit 0     =     3FH     ; Zero
Digit 1     =     06H     ; One
Digit 2     =     5BH     ; Two
Digit 3     =     4FH     ; Three
Digit 4     =     66H     ; Four
Digit 5     =     6DH     ; Five
Digit 6     =     7DH     ; Six
Digit 7     =     07H     ; Seven
Digit 8     =     7FH     ; Eight
Digit 9     =     67H     ; Nine
ENN         =     B7H     ; N
E           =     F9H     ; E
DASH        =     40H     ; -
ARE         =     D0H     ; R
TEE         =     F0H     ; T
```

8279 COMMANDS

```
CLKST       =     23H     ; Preset Clock to 100 KHZ
C8279       =     0DH     ; Select 8279 Control Port
D8279       =     0CH     ; Select 8279 Data Port
BLANKED     =     D0H     ; Clear Display RAM
DSPMOD      =     08H     ; Set display mode to 16
                            digit, left entry
DSPZER      =     90H     ; Display next character
                            in position zero
FIFRAM      =     40H     ; Data read from FIFO RAM
```

```
LOCATION    CODE      MNEMONIC                        COMMENTS

                          RESET ROUTINE

000         27        RESET:  CLR     A         ; Clear o/p Ports
001         39                MOVD    P1,A
002         3A                MOVD    P2,A
003         3F                MOVD    P7,A
004         B822              MOV     R0,#22H
006         27        CLMEM:  CLR     A         ; Clear Message Locations
007         A0                MOV     @R0,A
008         18                INC     R0
009         F8                MOV     A, R0
00A         5333              ANL     A,#33H
00C         9606              JNZ     CLMEM
00E         BF00              MOV     R7,#00H
010         BC01              MOV     R4,#01H
012         2323              MOV     A,#CLKST  ; Preset 8279 Clock
014         3400              CALL    PG8279
016         340B              CALL    BLANK     ; Clear Display
018         3441      START:  CALL    KBIN      ; Any Key Depressed?
01A         03F6              ADD     A,#F6H    ; Message to be transmitted?
01C         C6BA              JZ      OUTPUT
01E         03FD              ADD     A,#FDH    ; Is it "READ" Key?
020         C693              JZ      READ
022         03FF              ADD     A,#FFH    ; Is it "ENTER" Key?
024         C646              JZ      ENTER
```

```
026     03FF            ADD     A,#FFH   ; Is it "NUMBER" Key?
028     C62E            JZ      NUMBER
02A     0418            JMP     START
02C     0000

                "NUMBER" SUBROUTINE

02E     340B    NUMBER: CALL    BLANK
030     23B7            MOV     A,#ENN   ; Display " ⌐."
032     3417            CALL    DISPLAY
034     3441            CALL    KBIN     ; Any Key Depressed?
036     97              CLR     CARRY
037     03F6            ADD     A,#F6H   ; Message to be transmitted?
039     C6BA            JZ      OUTPUT
03B     F616            JC1     "START-2"; Invalid Number
03D     030A            ADD     A,#0AH
03F     AF              MOV     R7,A     ; Store Number
040     3421            CALL    DSPNR    ; Display It
042     0418            JMP     START
044     0000

                "ENTER" SUBROUTINE

046     340B    ENTER:  CALL    BLANK
048     23F9            MOV     A,#'E'   ; Display "E"
04A     3417            CALL    DISPLAY
04C     3441            CALL    KBIN     ; Any Key Depressed?
04E     97              CLR     CARRY
04F     03F6            ADD     A,#F6H   ; Message to be transmitted?
```

```
051     C6BA            JZ      OUTPUT
053     F616            JC1     "START-2"; Invalid Number
055     030A            ADD     A,#0AH
057     AA              MOV     R2,A        ; Store Temporarily
058     3421            CALL    DSPNR
05A     2340            MOV     A, #DASH ; Display "——"
05C     3417            CALL    DISPLAY
05E     FA              MOV     A,R2
05F     0321            ADD     A,#21H   ; Calculate Storage Address
061     A8              MOV     R0,A        ; Store It
062     BD00            MOV     R5,#00
064     3441    ENTR1:  CALL    KBIN        ; Any Key Depressed?
066     97              CLR     CARRY
067     03F6            ADD     A,#F6H   ; Message to be transmitted?
069     C6BA            JZ      OUTPUT
06B     F616            JC1     "START-2"; Invalid Message
06D     030A            ADD     A,#0AH
06F     530F            ANL     A,#0FH   ; Mask Higher Byte
071     47              SWAP    A
072     AA              MOV     R2,A        ; Save this Byte
073     47              SWAP    A
074     3421            CALL    DSPNR    ; Display It
076     3441            CALL    KBIN
078     97              CLR     CARRY
079     03F6            ADD     A,#F6H   ; Message to be transmitted?
07B     C6BA            JZ      OUTPUT
07D     F616            JC1     "START-2"; Invalid Message
```

```
07F     030A            ADD     A,#0AH
081     530F            ANL     A,#0FH      ; Mask Higher Byte
083     6A              ADD     A,R2
084     A0              MOV     @R0,A       ; Store This Byte
085     3421            CALL    DSPNR
087     2309            MOV     A,#09H
089     68              ADD     A,R0
08A     A8              MOV     R0,A        ; Storage Address of Next
                                                Byte
08B     FD              MOV     A,R5
08C     9618            JNZ     START
08E     1D              INC     R5
08F     0464            JMP     ENTR1
091     0000

                "READ" SUBROUTINE

093     340B    READ:   CALL    BLANK
095     23D0            MOV     A,#ARE      ; Display "⌐."
097     3417            CALL    DISPLAY
099     3441            CALL    KBIN        ; Any Key Depressed
09B     03F6            ADD     A,#F6H      ; Message to be transmitted?
09D     C6BA            JZ      OUTPUT
09F     030A            ADD     A,#0AH
0A1     AA              MOV     R2,A        ; Store Temporarily
0A2     3421            CALL    DSPNR       ; Display It
0A4     2340            MOV     A,#DASH     ; Display "___"
0A6     3417            CALL    DISPLAY
```

```
0A8     FA              MOV     A,R2
0A9     0321            ADD     A,#21H    ; Address of First Two Digits
0AB     A8              MOV     R0,A
0AC     F0              MOV     A,@R0     ; Display First Two Digits
0AD     3436            CALL    DSPREG
0AF     F8              MOV     A,R0
0B0     0309            ADD     A,#09H    ; Address of Last Two Digits
0B2     A8              MOV     R0,A
0B3     F0              MOV     A,@R0     ; Display Last Two Digits
0B4     3436            CALL    DSPREG
0B6     0418            JMP     START
0B8     0000
```

"OUTPUT" SUBROUTINE

```
BA      FF      OUTPUT: MOV     A,R7      ; No Message Stored?
BB      C618            JZ      START
BD      340B            CALL    BLANK     ; Clear Display
BF      23F0            MOV     A,#TEE    ; Display "⊢."
C1      3417            CALL    DISPLAY
C3      FC              MOV     A,R4
C4      A8              MOV     R0,A      ; Display Message Number
C5      3421            CALL    DSPNR
C7      2340            MOV     A,#DASH   ; Display "——"
C9      3417            CALL    DISPLAY
CB      F8              MOV     A, R0
CC      0321            ADD     A,#21H    ; Address of First Two
                                            Digits
```

```
CE      A8              MOV     R0,A
CF      F0              MOV     A, @R0
D0      39              MOVD    P1,A        ; Transmit First Two Digits
D1      3436            CALL    DSPREG      ; Display These
D3      F8              MOV     A,R0
D4      0309            ADD     A,#09H      ; Address of Last Two Digits
D6      A8              MOV     R0,A
D7      F0              MOV     A,@R0
D8      3A              MOVD    P2,A        ; Transmit Last Two Digits
D9      3F              MOVD    P7,A
DA      3436            CALL    DSPREG      ; Display These
DC      FC              MOV     A,R4
DD      37              CPL     A
DE      5F              ANL     A,R7
DF      C6E4            JZ      OUTPT1
E1      1C              INC     R4          ; Next Message Number
E2      0418            JMP     START
E4      BC01    OUTPT1: MOV     R4,#01H
E6      0418            JMP     START

                SUBROUTINE "PROGRAM 8279"

100     8A0D    PG8279: ORL     P2,#C8279; Select 8279 Control Port
102     91              MOVX    @R1,A       ; Send Command
103     81              MOVX    A,@R1       ; Read FIFO Status
104     F203            JB7     $-3         ; Transfer Complete?
106     9AF0            ANL     P2,#F0H     ; Deselect 8279
```

```
108     83              RET
109     0000

                SUBROUTINE "BLANK"

10B     23D0    BLANK:  MOV     A,#BLANKED ; Blank Display
10D     3400            CALL    PG8279
10F     2308            MOV     A,#DSPMOD  ; Set Display Mode
111     3400            CALL    PG8279
113     2390            MOV     A,#DSPZER  ; Set Display Position
                                           Zero
115     2400            JMP     PG8279

              SUBROUTINE "DISPLAY"

117     0AF0    DISPLAY:ANL     P2,#FOH
119     8A0C            ORL     P2,#0CH:   ; Select 8279 Data Port
11B     91              MOVX    @R1,A      ; Send Display Data
11C     9AF0            ANL     P2,#FOH    ; Deselect 8279
11E     83              RET
11F     0000

          SUBROUTINE "DISPLAY NUMBER"

121     530F    DISPNR: ANL     A,#0FH     ; Mask Higher Byte
123     032A            ADD     A,#DIGIT
125     A3              MOVP    A,@A
126     2417            JMP     DISPLAY
128     0000            NOP
12A     3F      DIGIT:  ZERO
```

```
12B     06              ONE
12C     5B              TWO
12D     4F              THREE
12E     66              FOUR
12F     6D              FIVE
130     7D              SIX
131     07              SEVEN
132     7F              EIGHT
133     67              NINE
134     0000            NOP

                SUBROUTINE "DISPLAY REGISTER"

136     AE      DSPREG: MOV     R6,A        ; Save it
137     47              SWAP    A
139     3421            CALL    DISPNR      ; Display Hi Digit
13A     FE              MOV     A,R6
13B     2421            JMP     DISPNR      ; Display Lo Digit
13D     0000            NOP
13F     0000            NOP

                SUBROUTINE "KEYBOARD INPUT"

141     2340    KBIN:   MOV     A,#FIFRAM   ; FIFO RAM READ Command
143     3400            CALL    PG8279
145     8A0D            ORL     P2,#0DH     ; Select 8279 Control Port
147     81              MOVX    A,@R1       ; Read FIFO Status
148     5307            ANL     A,#07H      ; Check Key Depression
```

```
14A     C655            JZ      FLGCHK
14C     9AF0            ANL     P2,#F0H
14E     8A0C            ORL     P2,#0CH  ; Select 8279 Data Port
150     81              MOVX    A,@R1
151     9AF0            ANL     P2,#F0H  ; Deselect 8279
153     83              RET
154     00              NOP
155     2641    FLGCHK: JNTO    KBIN     ; Check Transmit Flag
157     9AF0            ANL     P2,#F0H  ; Deselect 8279
159     230A            MOV     A,#0AH
15B     83              RET
```

CHAPTER V

ADVANTAGES OF MICROPROCESSOR VERSION OF THE DISTRESS ALARM KEYER

A hardwired version of the Distress Alarm Keyer was designed in 1977. A brief description of the hardwired logic design has been included in Chapter II. Its comparison has been made with the present software design and the advantages of the present design have been discussed in the first section. The second section includes a few ideas regarding auxiliary tasks which could be performed by the present design of the circuit.

1. Comparison of Hardwired and Software Designs

The hardwired design of the Distress Alarm Keyer was implemented using 15 Integrated Circuit chips (230 pins) whereas in the software design only 6 IC chips (147 pins) were used. If the software design was implemented without the display circuitry, the number of chips required would have been only 4 (112 pins). The lesser number of IC chips offers many advantages. Firstly, it increases the reliability of the circuit. It has been observed that the smaller the number of pins used in a circuit, the more reliable the circuit would be. Secondly, it makes the Distress Alarm Keyer more compact. It can be contained in a small box. The small physical size will make it possible for the user to mount it in any convenient place. The hardwired version of the Keyer was contained in two boxes since a previous application called for that design. Thirdly, the circuit wiring lay-out becomes very simple. The number of interconnecting wires is reduced to a great extent. This increases the reliability of

of the Keyer and contributes to the reduction of the circuit lay-out cost.

The present design of the Keyer includes an LED display which offers ease in operation. Any data entered from the keyboard is instantly displayed on the LEDs. If an invalid key is depressed, the data will neither be stored in the memory nor will it be displayed on the LEDs. The user would at once come to know his mistake. The data already stored in the memory can also be read at any instant. During the transmission of data to the Data Collection Platform, it would be displayed on the LEDs. The hardwired design of the Keyer had only a single indication of a key depression.

The software version of the Keyer offers a great deal of flexibility in the design and there is room for expansion of its functions. These auxiliary functions are discussed in the second section of this Chapter. There is no need to modify the circuit in order to achieve the enhancement of its performance. It can simply be accomplished by modifying the control program.

The only disadvantage of the present design is that it requires more power. The hardwired design was implemented using CMOS devices. Intel Corporation proposes to introduce a CMOS version of the 8748 microprocessor shortly. Using the CMOS devices power requirement of the circuit would reduce greatly.

2. Auxiliary Functions

NASA proposes to make the Satellite-Aided Search and Rescue system available to the general population. It would be used by persons flying aeroplanes, sailing in the sea, traveling on the road, and during various

expeditions. A reduced cost of the Distress Alarm Keyer and the Data Collection Platform would contribute towards making this system popular. The current design of the Keyer allows to incorporate additional functions in the system without any major modification in the circuit. The Keyer has been designed using a powerful microprocessor, Intel's 8748. Only one quarter of its 1-K byte of program memory has been used to perform the function of the Distress Alarm Keyer. The rest of the program memory can be utilized to perform other tasks.

As presently implemented, the Data Collection Platform is a separate unit. It receives the 16-bit word message from the Keyer and provides an unmodulated carrier, bit sync, frame sync, vessel I. D. Number, and transmits this data at 401.2 MHz $\pm$ 6 KHz in a one second burst every 64 seconds. Some of the data processing functions of the DCP could be incorporated in the current design of the Distress Alarm Keyer. Work is being done on the miniaturization of the DCP. The Keyer and the DCP would eventually be included in one package.

Another important function could be performed by using shock, heat, immersion or other sensors with the circuit, which would be continuously monitored by the microprocessor. In an event of emergency, the microprocessor would sense the signal and transmit the distress message if no human operator were able to do so.

When it is not being used as a Distress Alarm Keyer, the circuit can be utilized to perform other functions such as controlling other devices. It could also be programmed to be used as a clock or as a calculator capable of performing simple calculations.

LIST OF REFERENCES

LIST OF REFERENCES

1. "A Satellite-Aided Search and Rescue Program," NASA Goddard Space Flight Center, June 30, 1975.

2. Baker, J. L. "Sailboat Search and Rescue Experiment," Baker Development Corporation, Annapolis, Maryland, 1977.

3. "Intel MCS-48 Microcomputer User's Manual," Intel Corporation, Santa Clara, California, 1978.

4. "Location and Rescue Experiment and Demonstration," Baker Development Corporation, Annapolis, Maryland, July 22, 1975.

5. Richard, Dale and Satterlee, Paul, Jr. "Distress Alarm Keyer," Baker Development Corporation, Annapolis, Maryland, 1976.

APPENDICES

APPENDIX A

INTEL 8035 MICROPROCESSOR

The 8035 is an 8-bit microprocessor fabricated on a single chip using Intle's N-channel silicon-gate MOS process. It contains a 64x8 RAM data memory, 27 I/O lines, and an 8-bit timer/counter in addition to an on-board oscillator and clock circuits. It has an instruction set of 96 instructions most of which are single-byte instructions and no instruction is over two bytes in length. The Block Diagram of the microprocessor is shown in Figure A·1. The following sections describe the various functional blocks of the 8035.

1. Accumulator

It is the most important data register in the processor being one of the sources of input to the Arithmetic Logic Unit (ALU) and often the destination of the result of operations performed in the ALU. Data to and from I/O ports and memory also passes through the accumulator.

2. Arithmetic Logic Unit

The ALU accepts 8-bit data words from one or two sources and generates an 8-bit result under the control of Instruction Decoder.

3. Program Memory

The 8035 has no internal program memory and is used with external devices. There are three locations in the program memory of special importance:

Location "0"

Activating the Reset line of the processor causes the first

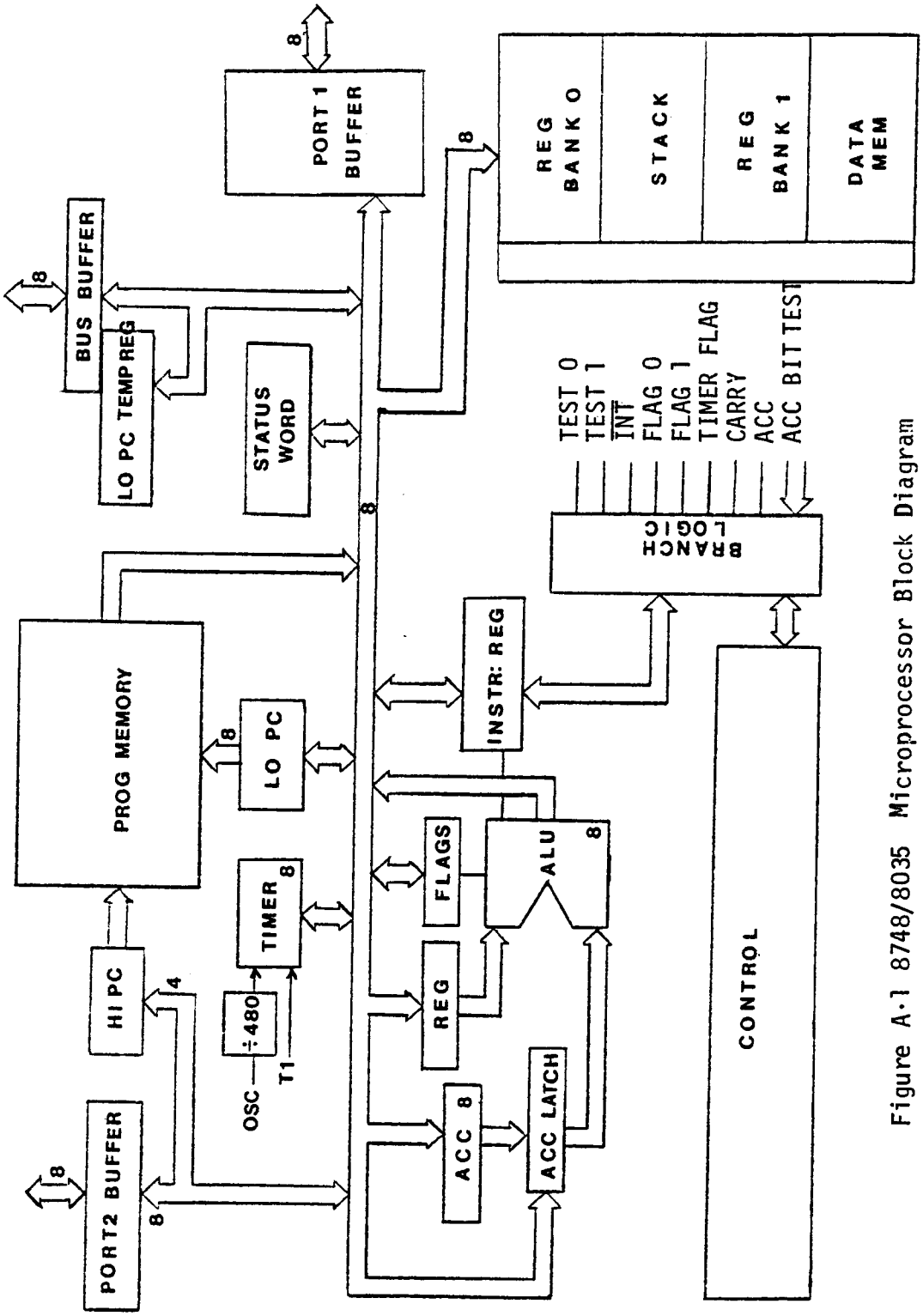

Figure A·1 8748/8035 Microprocessor Block Diagram

instruction to be fetched from location zero.

Location "3"

Activating the interrupt input line of the processor (if interrupt is enabled) causes a jump to subroutine. Therefore, the first word of an interrupt service routine is stored in location three.

Location "7"

A timer/counter interrupt resulting from timer/counter overflow (if enabled) causes a jump to subroutine. Therefore, the first word of a timer/counter service routine is stored in location seven.

4. Data Memory

Resident data memory is organized as 64 words 8-bits wide. It has two Register Banks, RB0 and RB1, either of which can be selected by executing (SEL RB) instruction. Each Register Bank consists of 8 working registers (R0-R7) which are directly addressable by several instructions. One of the register banks may be used as an extension of the other or reserved for use during subroutine. This allows the Register Bank used in main program to be instantly "saved" by a Bank Switch.

All locations in the data memory are indirectly addressable through R0 and R1 of the selected Register Bank. This allows the user to easily access up to four separate working areas in RAM at one time.

RAM locations (8-23) serve a dual role. These locations contain the program counter stack and are addressed by the Stack Pointer during subroutine calls as well as by RAM Pointer Registers R0 and R1. If the level of subroutine nesting is less than 8, all stack registers are not required and can be used as general purpose RAM locations.

5. Input/Output

The processor has 27 lines which can be used for input or output functions. These lines are grouped as 3 ports of 8 lines each which serve as either inputs, outputs or bidirectional ports and 3 tests inputs which can alter program sequences when tested by conditional jump instructions.

6. Ports 1 and 2

These ports are each 8 bits wide and have identical characteristics. Data written to these ports is statically latched and remains unchanged until written. As input ports these lines are non latching i.e. inputs must be present until read by an input instruction. Inputs are fully TTL compatible and outputs will drive one standard TTL load. Reset initializes all lines to the high impedance "1" state.

7. Bus

It is also an 8-bit port which is a true bidirectional port with associated input and output strobes. Input and output lines on this port cannot be mixed. As a static port, data is written and latched using the OUTL instruction and inputted using INS instruction. These instructions generate pulses on the corresponding $\overline{RD}$ and $\overline{WR}$ output strobe lines. As a bidirectional port the MOVX instructions are used to read and write the port. A write to the port generates a pulse on the $\overline{WR}$ output line and data is valid at the trailing edge of $\overline{WR}$. A read of the port generates a pulse on the $\overline{RD}$ output line and input data must be valid at the trailing edge of $\overline{RD}$. When not being written or read, the BUS lines are in a high impedance state.

8. Test and INT Inputs

Pins T0, T1 and $\overline{INT}$ serve as inputs and are testable with the conditional jump instruction.

9. Program Counter and Stack

The Program Counter is an independent counter which is used for Program Memory fetches. It is initialized to zero by activating the Reset line.

The Program counter stack is implemented using pairs of registers in the Data Memory Array. Data RAM locations 8 thru 23 are available as stack registers and are used to store the Program Counter and 4 bits of Program Status Word (Carry, Auxillary Carry, Flag 0, Register Bank Select). An interrupt or CALL to a subroutine causes these bits to be stored. The pair of registers to be used is determined by a 3-bit Stack Pointer which is part of the Program Status Word. Nesting of subroutines within subroutines can continue up to 8 times without overflowing the stack.

10. Program Status Word

The Program Status Word (PSW) is an 8-bit register which can be loaded to and from the accumulator. The PSW bit definitions are as follows:

Bit 0-2: Stack Pointer bits (S0, S1, S2)

Bit 3: Not used ("1" level when read)

Bit 4: Working Register Bank Switch bit (BS)

0 = Bank 0

1 = Bank 1

Bit 5: Flag 0 bit (F0); user controlled flag which can be complemented or cleared and tested with the conditional jump instruction JF0.

Bit 6: Auxiliary Carry (AC); carry bit generated by an ADD instruction and used by the decimal adjust instruction DA A.

Bit 7: Carry (CY); carry flag which indicates that the previous operation has resulted in overflow of the accumulator.

The upper four bits (Bit 4-Bit 7) of PSW are stored in the Program Counter Stack with every call to subroutine or interrupt vector and are optionally restored upon return with the RETR instruction. The RET return instruction does not update PSW.

11. Interrupt

An interrupt sequence is initiated by applying a low "0" level input to the $\overline{\text{INT}}$ pin. It is level triggered and active low to allow "WIRE-ORING" of several interrupt sources at the input pin. The interrupt line is sampled every machine cycle during ALE and when detected causes a "jump to subroutine" at location 3 to program memory as soon as all cycles of the current instruction are complete. The Program Counter and Program Status Word are saved in the stack. The end of an interrupt service routine is signalled by the execution of a Return and Restore Status instruction RETR. The interrupt system is single level in that once an interrupt is detected all further interrupt requests are ignored until execution of an RETR re-enables the interrupt input logic.

The interrupt input may be enabled or disabled under Program Control using the EN I and DIS I instructions. Interrupts are disabled by Reset and remain so until enabled by the user program.

12. Timer/Counter

The 8035 contains a counter to aid the user in counting external events and generating accurate time delays without placing a burden on the processor for these functions. In both modes the counter operation is the same, the only difference being the source of the input to the counter. The 8-bit up binary counter is presettable and readable with two MOV instructions. The counter content is not affected by Reset. Once started the counter will increment to its maximum count (FF) and overflow to zero. This results in the setting of an overflow flag flip-flop and in the generation of an interrupt request. The state of the flag in testable with the conditional jump instruction JTF. The flag is reset by executing a JTF or by Reset.

13. Clock and Timing Circuits

The clock and timing circuitry can be divided into the following functional blocks:

13.1 Oscillator

The microprocessor has an on-board oscillator which is a high-gain series-resonant circuit with a frequency range of 1 to 6 MHz. A crystal or inductor connected between X1 and X2 pins provides the feedback and phase shift required for oscillation. An externally generated clock may also be applied to X1-X2 as the frequency source.

13.2 State Counter

The output of the oscillator is divided by 3 in the State Counter to create a clock which defines the state times of the machine (CLK). CLK can be made available on the external pin T0 by executing an ENT0 CLK instruction. This output is disabled by Reset of the processor.

13.3 Cycle Counter

The Cycle Counter divides the CLK by 5 to provide a clock which defines a machine cycle consisting of 5 machine states. This clock is called Address Latch Enable (ALE) and is provided continuously on the ALE output pin.

14. Reset

The reset input provides a means for initialization for the microprocessor. This Schmitt-trigger input has an internal pullup resistor which in combination with an external 1 μF capacitor provides an internal reset pulse of sufficient length to guarantee all circuitry is reset. Reset performs the following functions:

1. Sets program counter to zero.
2. Sets stack pointer to zero.
3. Selects register bank zero.
4. Selects memory bank zero.
5. Sets BUS to high impedance state.
6. Sets Ports 1 and 2 to input mode.
7. Disables interrupts (timer and external).
8. Stops timer.
9. Clears timer flag.

10. Clears F0 and F1.

11. Disables clock output from T0.

15. Pin Description

The 8035 is packaged in a 40-pin Dual In-Line Package. The following is a summary of the functions of each pin. Each input is TTL compatible and each output will drive one standard TTL load.

Designation	Pin Number	Function
T0	1	Input pin testable using conditional jump instructions JT0 and JNT0. T0 can be designated as a clock output using ENT0 CLK instruction.
XTAL1	2	One side of crystal input for internal oscillator. Also input for external source.
XTAL2	3	Other side of crystal input.
$\overline{\text{RESET}}$	4	Input which is used to initialize the processor.
$\overline{\text{SS}}$	5	Single step input can be used in conjunction with ALE to "single step" the processor through each instruction. (Active Low).
$\overline{\text{INT}}$	6	Interrupt input. Initiates an interrupt if interrupt is enabled (Active Low).
EA	7	External Access input which forces all program memory fetches to reference external memory (Active High).

$\overline{RD}$	8	Output strobe activated during a BUS read. Can be used to enable data onto the BUS from an external device. (Active Low).
$\overline{PSEN}$	9	Program Store Enable. This output occurs only during a fetch to external program memory. (Active Low).
$\overline{WR}$	10	Output strobe during a BUS write. (Active Low).
ALE	11	Address Latch Enable. This signal occurs once during each cycle and is useful as a clock output. The negative edge of ALE strobes address into external data and program memory.
D0-D7 (BUS)	12-19	True bidirectional port which can be written or read synchronously using $\overline{RD}$, $\overline{WR}$ strobes. The port can also be statically latched. Contains the 8 low order program counter bits during program memory fetch and receives the addressed instruction under the control of PSEN.
Vss	20	Circuit ground potential.
P20-P27 (Port 2)	35-38	8-bit quasi-bidirectional port. P20-P23 contain the four high order program counter bits during memory fetch and serve as a 4-bit I/O expander bus for 8243.

PROG	25	Output strobe for 8243 I/O expander. Program pulse (+25V) input pin during 8748 programming.
VDD	26	Programming power supply; +25V during program, +5V during operation both ROM and PROM.
P10-P17 (Port 1)	27-34	8-bit quasi-bidirectional port.
T1	39	Input testable pin using JT1 and JNT1 instructions. Can be designated the event counter input using the STRT CNT instruction.
VCC	40	Main +5V power supply.

16. Instruction Set

The 8035 instruction set is extensive for a processor of its size and has been tailored to be straightforward and efficient in its use of program memory. All instructions are either one or two bytes in length and over 70% are only one byte long. Also, all instructions execute in either one or two cycles and over 50% of all instructions execute in a single cycle. Double cycle instructions include all immediate instructions, and all I/O instructions. Special instructions have also been included to simplify loop counters, table lookup routines, and N-way branch routines.

The following is the summary of the instruction set of the MCS-48 microcomputers.

Mnemonic	Description	Bytes	Cycle
ADD A, R	Add register to A	1	1
ADD A, @R	Add data memory to A	1	1
ADD A, #data	Add immediate to A	2	2
ADDC A, R	Add register with carry	1	1
ADDC A, @R	Add data memory with carry	1	1
ADDC A, #data	Add immediate with carry	2	2
ANL A, R	And register to A	1	1
ANL A, @R	And data memory to A	1	1
ANL A, #data	And immediate to A	2	2
ORL A, R	Or register to A	1	1
ORL A, @R	Or data memory to A	1	1
ORL A, #data	Or immediate to A	2	2
XRL A, R	Exclusive Or register to A	1	1
XRL A, @R	Exclusive Or data memory to A	1	1
XRL A, #data	Exclusive Or immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal Adjust A	1	1
SWAP A	Swap nibbles of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1

IN A, P	Input port to A	1	2
OUTL P, A	Output A to port	1	2
ANL P, #data	And immediate to port	2	2
ORL P, #data	Or immediate to port	2	2
INS A, BUS	Input BUS to A	1	2
OUTL BUS, A	Output A to BUS	1	2
ANL BUS, #data	And immediate to BUS	2	2
ORL BUS, #data	Or immediate to BUS	2	2
MOVD P, A	Output A to Expander port	1	2
ANLD P, A	And A to Expander port	1	2
ORLD P, A	Or A to Expander port	1	2
INC R	Increment register	1	1
INC @R	Increment data memory	1	1
DEC R	Decrement register	1	1
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ R, addr	Decrement register and skip	2	2
JC addr	Jump on Carry = 1	2	2
JNC addr	Jump on Carry = 0	2	2
JZ addr	Jump on A Zero	2	2
JNZ addr	Jump on A not Zero	2	2
JTO addr	Jump on TO = 1	2	2
JNTO addr	Jump on TO = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2

JF0 addr	Jump on F0 = 1	2	2
JF1 addr	Jump on F1 = 1	2	2
JTF addr	Jump on timer flag	2	2
JNI addr	Jump on $\overline{INT}$ = 0	2	2
JBb addr	Jump on Accumulator Bit	2	2
CALL	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2
CLR C	Clear Carry	1	1
CPL C	Complement Carry	1	1
CLR F0	Clear Flag 0	1	1
CPL F0	Complement Flag 0	1	1
CLR F1	Clear Flag 1	1	1
CPL F1	Complement Flag 1	1	1
MOV A, R	Move register to A	1	1
MOV A, @R	Move data memory to A	1	1
MOV A, #data	Move immediate to A	2	2
MOV R, A	Move A to register	1	1
MOV @R, A	Move A to data memory	1	1
MOV R, #data	Move immediate to register	2	2
MOV @R, #data	Move immediate to data memory	2	2
MOV A, PSW	Move PSW to A	1	1
MOV PSW, A	Move A to PSW	1	1
XCH A, R	Exchange A and register	1	1
XCH A, @R	Exchange A and data memory	1	1

XCHD A, @R	Exch. nibb. of A & data memory	1	1
MOVX A, @R	Move external data memory to A	1	2
MOVX @R, A	Move A to external data memory	1	2
MOVP A, @A	Move to A from current page	1	2
MOVP3 A, @A	Move to A from Page 3	1	2
MOV A, T	Read Timer/Counter	1	1
MOV T, A	Load Timer/Counter	1	1
STRT T	Start Timer	1	1
STRT CNT	Start Counter	1	1
STOP TCNT	Stop Timer/Counter	1	1
EN TCNTI	Enable Timer/Counter Interrupt	1	1
DIS TCNTI	Disable Timer/Counter Interrupt	1	1
EN I	Enable external interrupt	1	1
DIS I	Disable external interrupt	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
SEL MB0	Select memory bank 0	1	1
SEL MB1	Select memory bank 1	1	1
ENTO CLK	Enable Clock output on T0	1	1
NOP	No Operation	1	1

APPENDIX B

INTEL 8279 PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE

The 8279 is a general purpose programmable keyboard and display I/O interface device designed for use with Intel microprocessors. The keyboard portion can provide a scanned interface to a 64 contact key matrix which can be expanded to 128. It can also interface to any array of sensors or a strobed interface keyboard, such as the Hall effect and Ferrite variety. The display portion provides a scanned display interface for led, incandescent and other popular display technologies. Both right entry calculator and left entry typewriter display formats are possible.

1. Pin Description

The 8279 is packaged in 40-pin DIP. The following is a functional description of each pin:

Designation	No. of pins	Function
DB0-DB7	8	Bi-directional data bus. All data and commands between the CPU and the 8279 are transmitted on these lines.
CLK	1	Clock from system used to generate internal timing.
RESET	1	A high on this pin resets the 8279.
$\overline{CS}$	1	Chip Select. A low on this pin enables the interface functions to receive or transmit.

A0	1	Buffer Address. A high on this line indicates the signals in or out are interpreted as a command or status. A low indicates that they are data.
$\overline{RD}$, $\overline{WR}$	2	Input/Output read and write. These signals enable the data buffers to either send data to the external bus or receive it from the external bus.
IRQ	1	Interrupt Request. In a keyboard mode, the interrupt line is high when there is data in the FIFO/ Sensor RAM. The interrupt line goes low with each FIFO/Sensor RAM read and returns high if there is still information in the RAM.
V_{ss},V_{cc}	2	Ground and power supply pins.
SL0-SL3	4	Scan Lines which are used to scan the key switch or sensor matrix and the display digits. These lines can be either encoded (1 of 16) or decoded (1 of 4).
RL0-RL7	8	Return Line inputs which are connected to the scan lines through the keys or sensor switches. They have active internal pull-ups to keep .

		them high until a switch closure pulls one low.
SHIFT	1	The shift input status is stored along with the key position on key closure in the scanned keyboard modes. It has an active internal pull-up to keep it high until a switch closure pulls it low.
CNTL/STB	1	For keyboard modes this line is used as a control input and stored like status on a key closure. It has an active internal pull-up to keep it high until a switch closure pulls it low.
$OUTA_0$-$OUTA_3$	4	These two ports are the outputs for the 16x4 display refresh registers. The data from these outputs is synchronized to the scan lines for multiplexed digit displays. These two ports may also be considered as one 8 bit port.
$OUTB_0$-$OUTB_3$	4	
$\overline{BD}$	1	Blank Display. This output is used to blank the display during digit switching or by a display blanking command.

2. Principles of Operation

The following is a description of the major elements of the 8279 chip. Refer to the block diagram in Figure B. 1.

2.1 I/O Control and Data Buffers

The I/O control section uses $\overline{CS}$, A0, $\overline{RD}$ and $\overline{WR}$ lines to control data flow to and from the various internal registers and buffers. The Data Buffers are bi-directional buffers that connect the internal bus to the external bus. When the chip is not selected ($\overline{CS}$ = 1), the devices are in high impedance state. The drivers input during $WR \cdot \overline{CS}$ and output during $RD \cdot \overline{CS}$. The source of data read is specified by the Read FIFO or Read Display commands. The data is always written to the Display RAM. The address is specified by the latest Read Display or Write Display command.

2.2 Control and Timing Registers and Timing Control

These registers store the keyboard and display modes and other operating conditions programmed by the CPU. The timing control contains the basic timing counter chain. The first counter is a divide by "N" prescalar that can be programmed to match the CPU cycle time to the internal timing. The prescalar is software programmed to a value between 2 and 31. A value which yields an internal frequency of 100 KHz gives a 5.1-ms keyboard scan time and 10.3-ms debounce time. The other counters divide down the basic internal frequency to provide the proper key scan, row scan, keyboard matrix scan, and display scan lines.

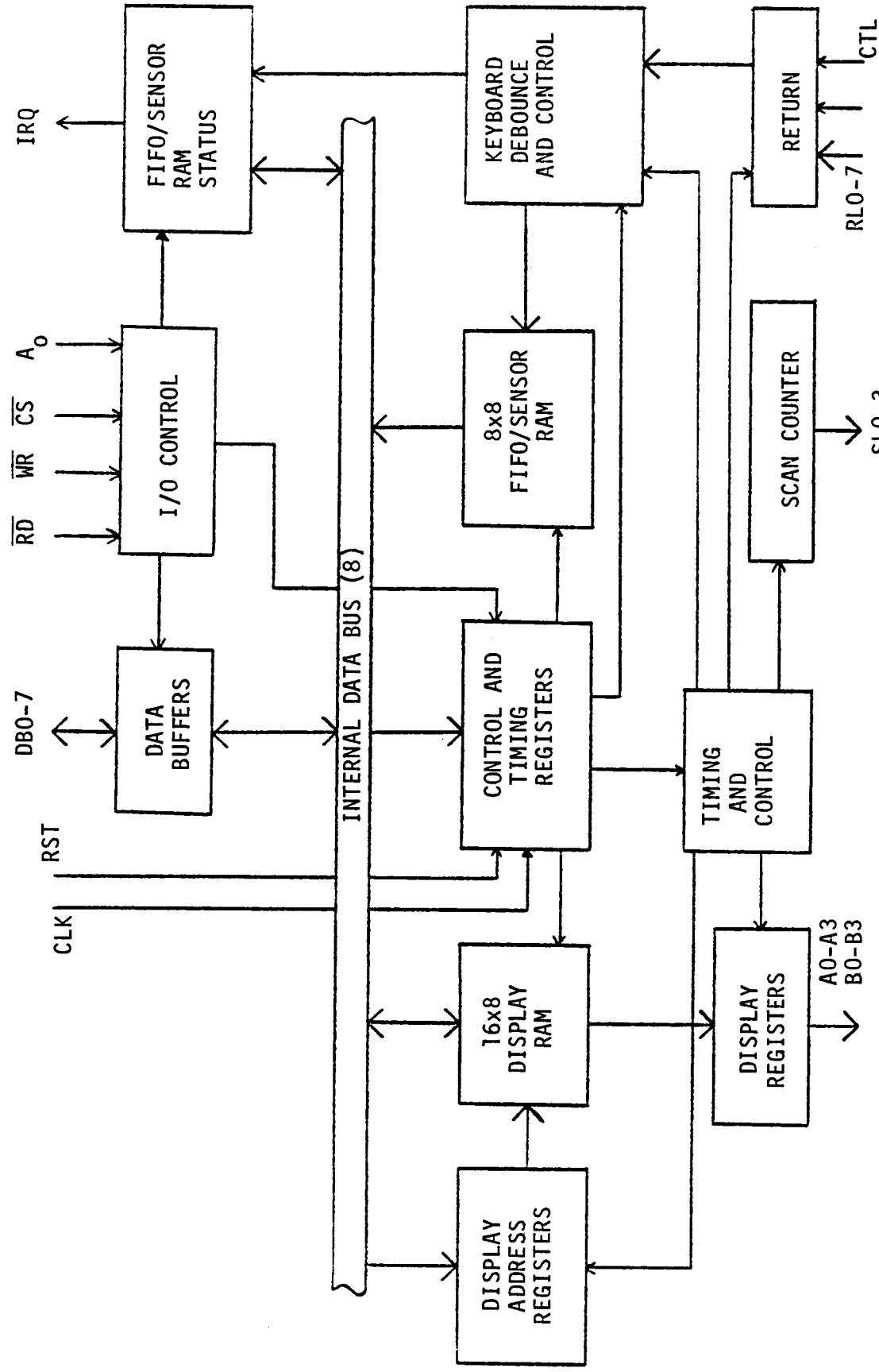

Figure B·1 8279 Block Diagram

2.3 Scan Counter

The scan counter has two modes. In the encoded mode, the counter provides a binary count that must be externally decoded to provide the scan lines for the keyboard and display. In the decoded mode, the scan counter decodes the least significant two bits and provides a decoded 1 of 4 scan. In the encoded mode, the scan lines are active high outputs. In the decoded mode, the scan lines are active low outputs.

2.4 Return Buffers and Keyboard Debounce and Control

The 8 return lines are buffered and latched by the Return Buffers. In the keyboard mode, these lines are scanned, looking for key closures in that row. If the debounce circuit detects a close switch, it waits about 10 msec to check if the switch remain closed. If it does, the address of the switch in the matrix plus the the status of SHIFT and CONTROL are transferred to the FIFO. The data format is as shown in Figure B.2.

In the scanned Sensor Matrix modes, the contents of the return lines are directly transferred to the corresponding row of the Sensor RAM(FIFO) each key scan time. In strobed input mode, the contents of the return lines are transferred to the FIFO on the rising edge of the CNTL/STB line pulse. The data format for scanned Sensor Matrix and Strobbed Input modes is shown in Figure B.3.

2.5 FIFO/Sensor RAM and Status

FIFO/Sensor RAM is a dual function 8x8 RAM. In Keyboard or Strobed Input modes, it is a FIFO. Each new entry is written into successive RAM positions and each is then read in order of entry. FIFO status

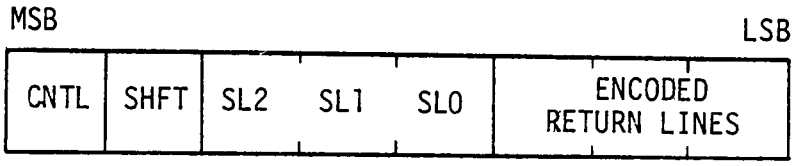

Figure B·2 Scanned Keyboard Data Format

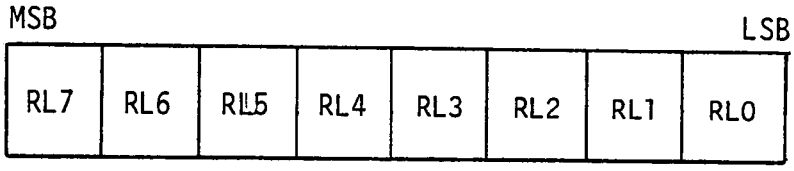

Figure B·3 Strobbed/Sensor Matrix Data Format

keeps track of the number of characters in the FIFO. Too many reads or writes will be recognized as an error. There are two types of errors possible: overrun and underrun. Overrun occurs when the entry of another character into a full FIFO is attempted. Underrun occurs when the CPU tries to read an empty FIFO.

In Scanned Sensor Matrix mode, the memory is a Sensor RAM. Each row of the Sensor RAM is loaded with the status of the corresponding row of sensor in the sensor matrix. In this mode, IRQ is high if a change in a sensor is detected. A bit is set in the FIFO status word to indicate that at least one sensor closure indication is contained in the Sensor RAM.

The FIFO status word also has a bit to indicate that the Display RAM was unavailable because a Clear Display or Clear All command had not completed its clearing operation. In Special Error Mode the S/E bit is showing the error flag and serves as an indication to whether a simultaneous multiple closure error has occured. The following is a description of the bits of the FIFO Status Word.

Bit 0-2(NNN)	Number of characters in FIFO
Bit 3 (F)	FIFO full
Bit 4 (U)	Error-Underrun
Bit 5 (O)	Error-Overrun
Bit 6 (S/E)	Sensor Closure/Error flag for multiple closure
Bit 7 (D_u)	Display unavailable.

2.6 Display Address Registers and Display RAM

The Display Address Register holds the address of the word currently being written or read by the CPU and the two 4-bit nibbles being

displayed. The read/write addresses are programmed by CPU command. The Display RAM can be directly read by the CPU. Data entry to the display can be set to either left or right entry.

3. Software Operation

The following commands program the 8279 operating modes. The commands are sent on the Data Bus with $\overline{CS}$ low and A0 high and are loaded to the 8279 on the rising edge of $\overline{WR}$.

3.1 Keyboard/Display Mode Set

	MSB							LSB
Code:	0	0	0	D	D	K	K	K

Where DD is the Display Mode and KKK is the keyboard Mode.

DD

00	8 8-bit character display-Left entry
01	16 8-bit character display-Left entry
10	8 8-bit character display Right entry
11	16 8-bit character display-Right entry

Note that when decoded scan is set in keyboard mode, the display is reduced to 4 characters independent of display mode set.

KKK

000	Encoded Scan Keyboard——2 Key Lockout
001	Decoded Scan Keyboard——2 Key Lockout
010	Encoded Scan Keyboard——N-Key Rollover
011	Decoded Scan Keyboard——N-Key Rollover
100	Encoded Scan Sensor Matrix
101	Decoded Scan Sensor Matrix

110 Strobed Input, Encoded Display Scan

111 Strobed Input, Decoded Display Scan

3.2 Program Clock

	MSB							LSB
Code:	0	0	1	P	P	P	P	P

Where PPPPP is the prescaler value 2 to 31. The programmable prescaler divides the external clock by PPPPP to get the basic internal frequency. Default after a reset pulse is 31.

3.3 Read FIFO/Sensor RAM

	MSB							LSB
Code:	0	1	0	A1	X	A	A	A

Where A1 is the Auto-increment flag for the Sensor RAM and AAA is the row that is going to be read by the CPU. A1 and AAA are used only if the mode is set to Sensor Matrix. In the Auto-Increment mode for reading data from the FIFO/Sensor RAM, each read advances the address by one so that the next read is from the next character.

3.4 Read Display RAM

	MSB							LSB
Code:	0	1	1	A1	A	A	A	A

Where A1 is the Auto-Increment flag for the Display RAM and AAAA is the character that the CPU is going to read next. Since the CPU uses the same counter for reading and writing, this command also sets the

next write location and Auto-Increment mode. This command is used to specify the display RAM as the data source for CPU data reads.

3.5 Write Display RAM

	MSB							LSB
Code:	1	0	0	A1	A	A	A	A

Where A1 is the Auto-Increment flag for the Display RAM and AAAA is the character that the CPU is going to write next. The addressing and Auto-Increment are identical to Read Display RAM. The CPU will read from whichever RAM (Display or FIFO/Sensor) was last specified.

3.6 Display Write Inhibit/Blanking

Code:	1	0	1	X	IW	IW	BL	BL
					A	B	A	B

Where IW is Inhibit Writing (nibble A or B) and BL is Blanking (nibble A or B). If the display is being used as a dual 4-bit display, then it is necessary to mask one of the 4-bit halves so that entries to the Display from the CPU do not affect the other half. The IW flags allow the programmer to do this. The BL flags blank the display. A "1" sets the flag. Reissuing the command with a "0" resets the flag.

3.7 Clear

	LSB							LSB
Code:	1	1	0	CD	CD	CD	CF	CA

Where CD is Clear Display, CF is Clear FIFO Status (including interrupt), and CA is Clear All. CD is used to clear all positions of the Display RAM to a programmable code. Clearing the display takes one display scan. CF sets the FIFO status to empty and resets the interrupt output line. CA has the combined effect of CD and CF. CA uses the CD clearing code to determine how to clear the Display RAM. CA also resets the internal timing chain to resynchronize it.

3.8 End Interrupt/Error Mode Set

	MSB							LSB
Code:	1	1	1	E	X	X	X	X

For the sensor matrix mode this command lowers the IRQ line and enables further writing into RAM. (The IRQ line would have been raised upon the detection of a change in a sensor value. This would have also inhibited further writing into the RAM until reset.)

For the N-Key rollover mode, if E bit is programmed to "1" the chip will operate in the special Error mode.

VITA

Rashid Naseer Khan was born in Lieah, Pakistan as the fourth child of a family with five children. His father Nasir Uddin Khan, a practicing lawyer and his mother a housewife lives in the city of Lahore, a historical and educational center of Pakistan. Rashid received his M.S. in Electrical Engineering from the University of Tennessee in August, 1979.

He attended Cadet College in Hasan Abdal and graduated from there in 1967. During his stay there, he was awarded a merit scholarship for outstanding academic achievements. He was admitted to the University of Engineering and Technology, Lahore from where he obtained his B.S. in 1971.

In April, 1972 Rashid was appointed as a Television Engineer in Pakistan Television Corporation. He worked there until December, 1974, when was selected for an overseas assignment with Libyan Broadcasting Corporation. He was admitted to the University of Tennessee, Knoxville in September, 1977.

Rashid has always been a keen sportsman and actively participated in many extra-curricular activities. He was a member of the Honor Society of PHI KAPPA PHI.