12-1985

Direct Memory Acquisition of Fast Analog Signals

Syed Javaid Iqbal

DIRECT MEMORY ACQUISITION OF FAST ANALOG SIGNALS

by

Syed Javaid Iqbal

A Thesis
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Master of Arts
Department of Physics

Western Michigan University
Kalamazoo, Michigan
December 1985

ACKNOWLEDGEMENTS

My sincere appreciation is extended to Dr. John Tanis, Dr. Eugene Bernstein, Dr. Gerald Hardie, Dr. Subramanian Ganesan, Dr. Larry Oppliger, and Dr. Michitoshi Soga. Their advice and assistance with the implementation of this project and with the writing of this thesis were absolutely necessary and important.

I am especially indebted to Dr. John Tanis, adviser, for his time, encouragement, understanding, and assistance with the writing of this thesis. Without his extensive efforts this thesis would not have been completed.

A particular note of appreciation goes to Dr. Bernstein, chairman of the physics department, for allowing me to use his physics laboratory.

Special thanks goes to my friends Miss Norma Awang Hard and Miss Kay Pedersen for their love, encouragement, and patience during the years of my study.

This thesis is dedicated to my sister, Syeda Naheed Sultana, who died on March 21, 1983, during my study period at Western Michigan University.

Syed Javaid Iqbal

DIRECT MEMORY ACQUISITION OF FAST ANALOG SIGNALS

Syed Javaid Iqbal, M.A.

Western Michigan University, 1985

A fast data transfer computer circuit is developed using the direct memory acquisition (DMAC) technique. For this purpose Z-80 microcomputer chips, manufactured by Ziolog, were used. The direct memory acquisition circuitry was developed in several stages and required both hardware and software development. An erasable programmable read only memory (EPROM) programmer was built for reading and loading the system program. The design, construction, and testing of the DMAC system are discussed in detail and an example of the use of this system for collecting data is given.

TABLE OF CONTENTS

Table of Contents-continued

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The year 1971 marked the dawn of a new age in integrated circuit technology. In that year Intel corporation took advantage of large scale integrated circuit (LSI) technology and introduced the world's first microprocessor chip, a 4-bit computer named the 4004. In 1973 the 4-bit machine was replaced by a newer 8-bit microprocessor, the Intel 8008. In the next few years the semiconductor industry introduced literally dozens of microprocessor circuits, each faster and more complicated than its predecessor. In 1976 the Z-80 Central Processing Unit (CPU) and its supporting chips were introduced by Zilog, representing the state-of-the-art in 8-bit microprocessors. Zilog is currently developing a successor to the Z-80 line, the Z-80000 series consisting of a CPU and several support chips. However, the Z-80000 series will be a 32-bit CPU with a computational capacity comparable to medium sized computers thereby representing a significant jump in capability for microcomputers. Additional capabilities in microcomputers will be manifested in very fast, mass memory systems. Since 32-bit microprocessors, which can address up to 4-billion memory locations, and megabyte bubble memories are now

available, new computers must be very fast to take full advantage of these chips.

In this thesis a very high speed data transfer technique will be discussed. Presently about 95% of CPU time is used in data transfer, from memory to memory, input/output (I/O) to input/output, memory to I/O or I/O to memory. To speed data acquisition the direct memory access technique is used. In this technique data do not go through the CPU but are transferred directly to memory using a chip known as a Direct Memory Access Control (DMAC). In the work described here, the data transfer rate is 1.2 million bytes per second, which is about 100 times faster than that of the Western Michigan University DEC-10 system. Nowadays scientists are using this technique in sound pattern recognition, digital communication and satellite communication. This technique will also be used in fiber optics communication. In libraries computer applications of fast data acquisition will be very important as students try to access thousands of books at the same time.

The microprocessor is a complicated chip and it has many different sections and each section performs a specific task. A brief description of a microprocessor and its parts is given in Chapter II. In the work described here a Z-80 microprocessor is used. This microprocessor is explained in detail in the third chapter

in which a functional block diagram of the Z-80 CPU, the main register set, and the alternate register set are shown along with a pin diagram of the Z-80 and a brief description of each pin.

For all types of computers, the designer has to provide system programs (software) which tell the hardware system what to do and how to do it. Once the software is developed it has to be loaded into the system. These topics are discussed in Chapter IV. In the present work an Erasable Programable Read Only (EPROM) chip was used to provide the system software and so it was necessary to design and build an EPROM programmer.

In order to design a fast memory acquisition circuit several steps were required. During each step test circuits and programs (software) were used to show that everything was working properly up to that stage. The test circuits are described in Chapter V and the programs are attached as appendices.

In many computer systems the memory which is most often accessed by the end user is the Random Access Memory (RAM). Storing the data in RAM and interfacing it with the direct memory access controller (DMAC) are discussed in the final chapter. A system program, written in machine language, to control the data acquisition is also attached as Appendix E.

CHAPTER II

MICROPROCESSOR BASICS

The microprocessor is the basic building block of any microcomputer. If we examine a microprocessor carefully, we find that it consists of five major sections: the arithmetic logic unit (ALU), storage (memory), control, input and output. The ALU performs arithmetic operations, such as addition, subtraction, multiplication, division and other logical operations. The control section regulates the transfer of data among the other sections. The storage section contains the program, i.e., the sequence of computer instructions that have been designed to perform certain tasks, as well as intermediate results of computations. The input section provides a means to enter information to be processed by the computer. This section varies technically from one computer to another, but it is that section which accepts input from external devices and transmits this information to the machine. If the computer has no way to communicate information to the outside world, then its computations are useless. This capability is provided by the output section, which also varies technically from one computer to another, but it is that section of the computer which transmits the computed results to some external device. The complete computer

can accept input data, process it, and output the results.

The term microprocessor means a single large scale integrated (LSI) chip which contains the ALU and control functions of a computer. Program and data storage, and the input/output functions are performed by other support chips designed especially for one particular microprocessor.

Most computers are built around a microprocessor and are known as microcomputers. The microcomputer is a collection of ICs instead of a single IC and is usually assembled on a printed circuit (PC) board. Though the microcomputer on a printed circuit board is a complete computer, it still requires other devices vital to its operation, for example, a power supply to provide power for operation. This aspect is of little relevance in an overall discussion of microprocessor architecture, but it is important to note that these peripheral devices may exceed the cost of the entire computer.

Finally there must be a means for communication with a human operator so that data may be put into the computer and results obtained from it. This I/O communication capability provides a complete computer system.

Memory Addressing and Memory Types

In any computer system data must be transferred between the storage section of the computer and the arithmetic logic unit. The technique for selecting data to be transferred to or from memory is called addressing. When the processor requires data from memory, it places the particular address on the address bus (bus is the term for a path of data bits). The memory responds to the request, if activated, by placing the appropriate word on the data bus. A similar technique is used for storing data into memory. First the microprocessor places the address where the data is to be stored on the address bus and then the data itself on the bus.

There are four primary classes of memory, read-only memory (ROM), random access memory (RAM), eraseable programmable read-only memory (EPROM), and electrically erasable programmable read-only memory (EEPROM).

A read-only memory does not allow the microprocessor to store data in those memory locations. This memory is typically used to store program instructions and other unchangeable data. There are many subspecies of read-only memories. One type of read-only memory is a mask programmable read-only memory (ROM). This refers to a single integrated circuit that is programmed by a manufacturer during the construction process. This is

only used for high volume applications.

Programmable read-only memory (PROM) is a memory of the same class, but this memory is progammed by the end user. A PROM is programmed by "burning out" fused links in those cells which are to contain a data zero or one, depending on the particular PROM.

In a random access memory, the user is allowed to read or write. This is done by the processor. First a particular address is presented on the address bus, the read line is activated and then the CPU responds to the request by presenting data at that address. The same process is used for writing with the exception that, this time the write line is activated. Whenever a processor writes data to a memory location the old data is deleted and the new data remains in the memory as long as the computer is switched on.

In an EPROM the memory is programmed by the user and it can be erased by exposing it to ultraviolet light. All data at all addresses are erased simultanously.

An EEPROM is also programmed by the end user, but the data at any particular address can be electrically erased. This option is not available in any other kind of memory.

Registers

A register is the basic digital building block of any CPU. A computer can be visualized as a set of registers and pathways between the registers. A description of the computer's registers provides a great deal of insight into the capabilities of the microprocessor. There are several types of registers within a microprocessor. An accumulator register is dedicated to computational tasks. This means that the arithmetic logic unit can use this register as an operand and can store its results there. There is another type of register called an index register. An index register is used to provide address selection and incrementing within memory. When the program refers to a particular area of memory, the index register can provide the address of the particular memory location to be accessed. Every computer also contains a special purpose register. This is the program counter (PC). This register contains the address of the next instruction that the computer is to execute. After executing that instruction the PC register is updated. This instruction itself is in memory. In other words the program counter register is a marker for the microprocessor so that it can "remember" which instruction is to be performed next. The number of bits contained in the program counter register determines the amount of

memory that can directly be accessed by the microprocessor.

CHAPTER III

THE Z-80 MICROPROCESSOR

We have discussed microprocessors in general in the last chapter. Here we shall discuss the Z-80 microprocessor in detail. First we shall look at a functional diagram of the Z-80 microprocessor chip and then we shall discuss the pin configuration of the 40-pin dual in-line package (DIP) which comprises the Z-80 electronic device. Figure 1 is the functional block diagram of the Z-80 CPU (Components Data Book, 1984).

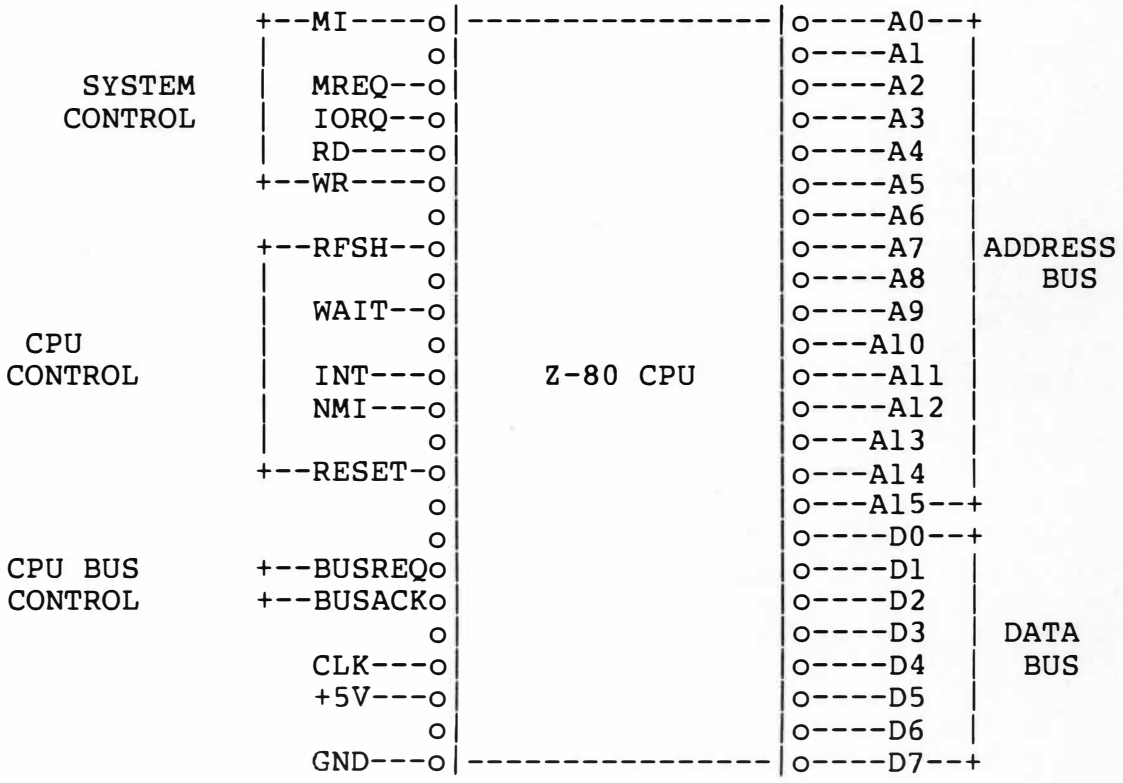


Figure 1. Z-80 CPU

As the computer executes a program residing in its external memory, each sequential instruction is read from memory by placing the address contained in the program counter (PC) register on the address bus at the same time it generates proper control signals on the control bus to activate the memory, and then reads the data on the data bus into the proper register within the CPU. Timing is critical to ensure that the addressed memory contents of the location are on the data bus when the CPU reads the data bus. The CPU control lines coordinate these tasks and ensure that instruction operation codes placed in the instruction register are properly decoded. The CPU timing also controls the arithmetic logic unit (ALU) in performing all the arithmetic and logic operations supported by the Z-80 instruction set. Operations include add, subtract, logical AND, logical OR, logical exclusive OR, compare, left or right shift and rotates, increment, decrement, set bit, and test bit.

The ALU, in performing these operations, communicates via the internal data bus, shown in the diagram, with the 22 internal registers, the instruction register, and the data bus controller. The controllers for the data and address buses oversee all the activities relating to the exchange of data between the CPU and the external world via their respective buses. The data bus is bidirectional while the address bus is unidirectional. No data is

received by the CPU on the address bus. A diagram of the configurations of the CPU registers are shown in Figure 2, Figure 3 and Figure 4 (Components Data Book, 1984).

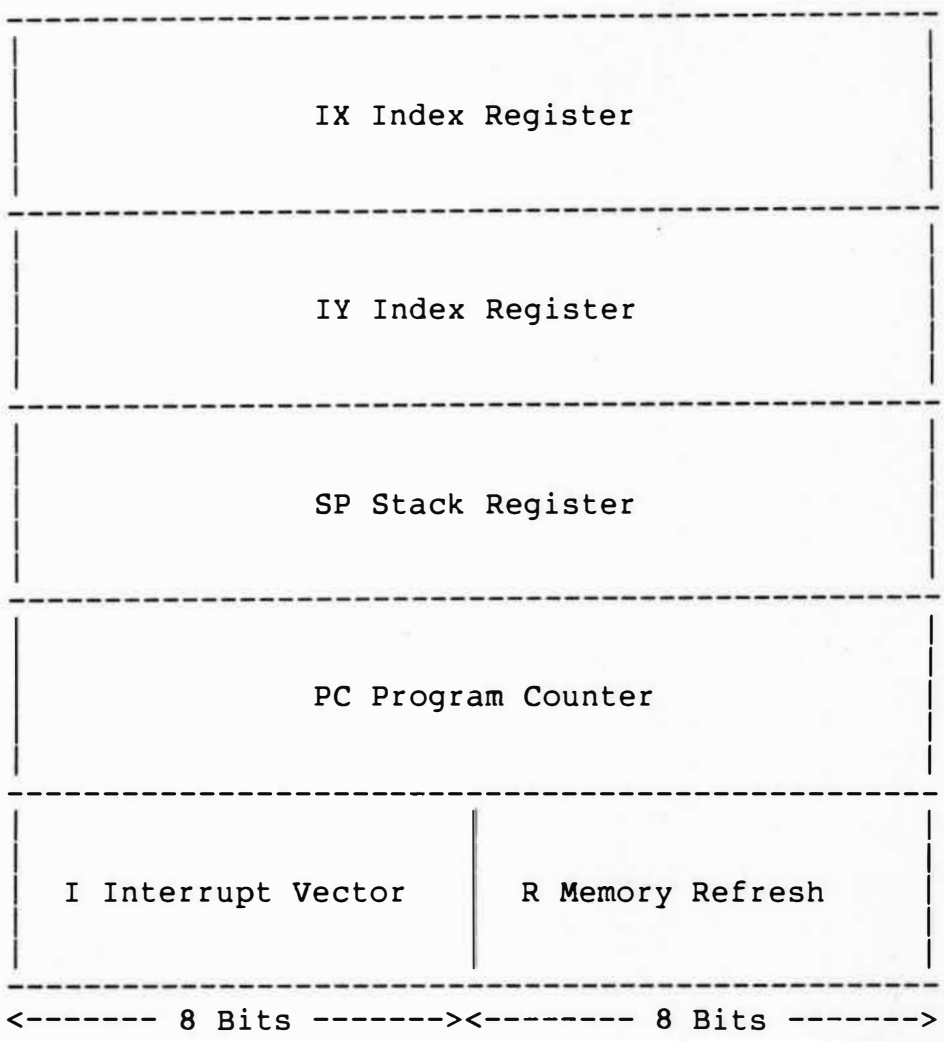
A Accumulator	F Flag Register
B General Purpose	C General Purpose
D General Purpose	E General Purpose
H General Purpose	L General Purpose

<-------- 8 Bits -------><------- 8 Bits ------->

Figure 2. Main Registers-Set A

A' Accumulator	F' Flag Register
B' General Purpose	C' General Purpose
D' General Purpose	E' General Purpose
H' General Purpose	L' General Purpose

Figure 3. Alternate Register Set



The Pin Functions of the Z-80 CPU

The pin functions for the Z-80 CPU are shown in Figure 5. These functions are briefly discussed below.

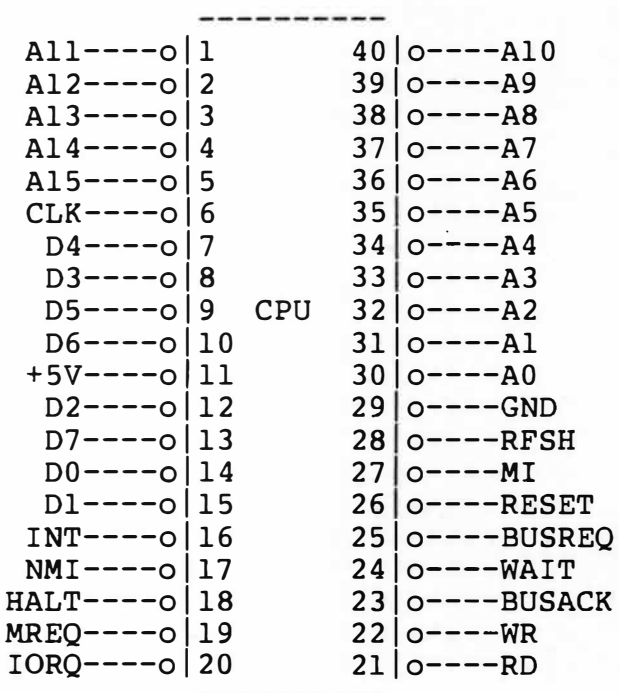


Figure 5. Central Processing Unit

A0-A15 (address bus)

Tri-state (high, low, and high impedance states), output, active high. A0-A15 constitute a 16-bit address. The address bus provides the address for memory (up to 65536 bytes), data exchanges, and for I/O device data exchanges. I/O addressing uses the eight lower address bits to allow direct selection of up to 256 input or 256 output ports. Since four addresses are required for each I/O device the total number of I/O ports which can be directly connected to the Z-80 CPU are 64.

D0-D7 (data bus)

Tristate, input/output, active high, 8-bit bidirectional data bus. The data bus is used for data exchange with memory and I/O devices.

MI (machine cycle one)

Output, active low. MI indicates that the current machine cycle is the op-code fetch cycle of an instruction execution. During 2-byte execution op-codes, MI is generated as each op-code is fetched. These two byte codes begin with CB, DD, ED, or FD (hex). MI also occurs with IORQ to indicate an interrupt acknowledge cycle.

MREQ (memory request)

Tri-state, output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

IORQ (input/output request)

Tri-state, output, active low. The Input/Output Request signal indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. An IORQ signal is also generated with an MI signal when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed

on the data bus. Interrupt acknowledge operations occur during MI time.

RD (memory read)

Tri-state, output, active low. RD indicates that the microprocessor wants to read data from memory or an I/O device. The addressed memory or I/O device should use this signal to gate data onto the CPU data bus.

WR (memory write)

Tri-state, output, active low. WR indicates that the CPU data bus is holding valid data to be stored in the addressed I/O or memory device.

RFSH (refresh memory)

Output, active low. RFSH indicates that the lower seven bits of the address bus contain a refresh address for dynamic memories and the current MREQ signal should be used to do a refresh read to all dynamic memories.

HALT (halt state)

Output, active low. Halt indicates that the CPU has just executed a halt instruction and is waiting for an interrupt before operation can resume. While halted, the CPU executes NOPs (no operation) instructions to maintain memory refresh.

WAIT (wait)

Input, active low. WAIT indicates to the Z-80 CPU that the addressed memory or I/O devices are not ready for data transfer. The CPU continues to enter wait states for as long as this signal is active. This signal allows memories of I/O devices of any speed to be synchronized to the CPU.

INT (interrupt request)

Input, active low. The interrupt request is generated by the external device. A signal will be acknowledged at the end of the current instruction if the BUSRQ signal is not active and the internal software controlled interrupt enable flip-flop (IFF) is enabled.

NMI (non-maskable interrupt)

Input, negative-edge triggered. NMI has a higher priority than INT and is always recognized at the end of the current machine cycle.

RESET (reset)

Input, active low. RESET forces the program counter to zero and initilizes the CPU. During the reset time, the address bus and data bus go to a high impedance state and all control output goes to the inactive state.

BUSRQ (bus request)

Input, active low. The BUSRQ requests the CPU data bus, the address bus, and the tri-state output control signal to go to a high impedance state so that other devices can control these buses. Usually this is used by a DMAC (direct memory access control).

BUSACK (bus acknowledge)

Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, the data bus and the tri-state status are to be controlled by the external device.

Z-80 CPU Timing Diagrams

Z-80 CPU timing diagrams are given in Appendix A. These diagrams show the sequence of the events which are allowed to take place. The timing sequence is one of the most important aspects of any microprocessor.

CHAPTER IV

EPROM PROGRAMMER

The performance of a computer depends on its hardware as well as software. These two things are interdependent. In order to write a system software program it is necessary to know how the control lines are connected to the external devices. System software tells the CPU what to do and how to do it. This program is always written in a PROM or ROM.

In this work an Erasable Programmable Read-Only Memory (EPROM) was used to provide the system software because it is easy to program and mistakes are easily corrected by reprogramming. For programming the EPROM, a circuit called an EPROM Programmer was developed. A block diagram of the EPROM programmer is given below.

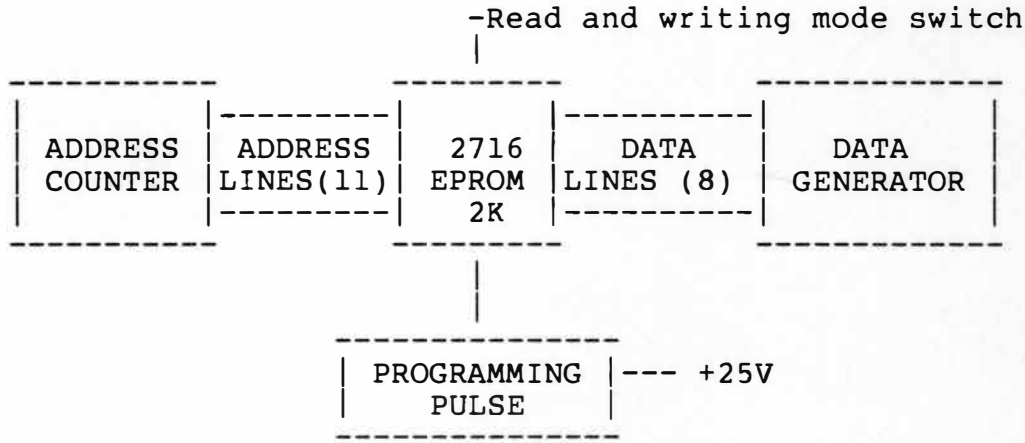


Figure 6. EPROM Programmer

The EPROM Programmer can be divided into three main sections, address counter, data counter, and programming pulse generator.

Address Counter

In order to program the EPROM, first the address bus is activated. A block diagram of the address counter is given below.

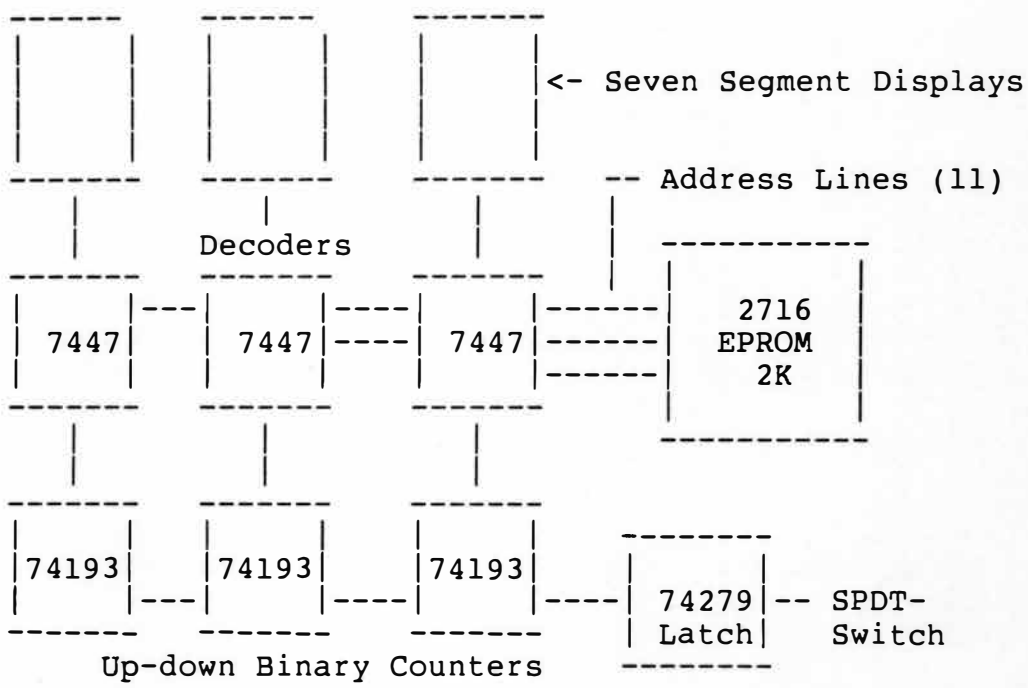


Figure 7. Address Counter

In this circuit three different kinds of ICs are used. The (74279) is a latch which is used to elminate "bounce" (i.e., make and break in the connection due to mechanical *vibration) in mechanical switches. The up-down binary*

counter (74193) is used for counting signal pulses and the BCD decoders (7447) are used to drive seven segement displays to show the selected address.

A signal is generated by pushing the SPDT (single pole double throw) switch. First this signal is latched and then it is counted by the binary counter. Data are displayed with seven segment displays to make sure that the correct address is on the address bus.

Each seven segment display can display digits from 0 to 15 (0 to F). After counting to 15, the binary counter generates a CARRY before receiving the next signal. This carry is counted by the next binary counter and also resets the first binary counter. This procedure is repeated until the secound counter also generates a CARRY, which resets both counters. The same procedure is repeated for the third counter and so on. Three binary counters were used in the present work, meaning 4K could be addressed. This circuit can easily be expanded up to 64K by just adding one more counter.

Since the 74193 binary counter can count up as well as down the SPDT switch can be used in its second position to count down. Hence the address can count up from 0 to 4K or down from 4K to 0. During reading or writing the address bus is connected directly to the address bus of the EPROM.

Data Generator

Once the address bus is activated, data must be generated on the data bus. A block diagram of the data generator is shown below.

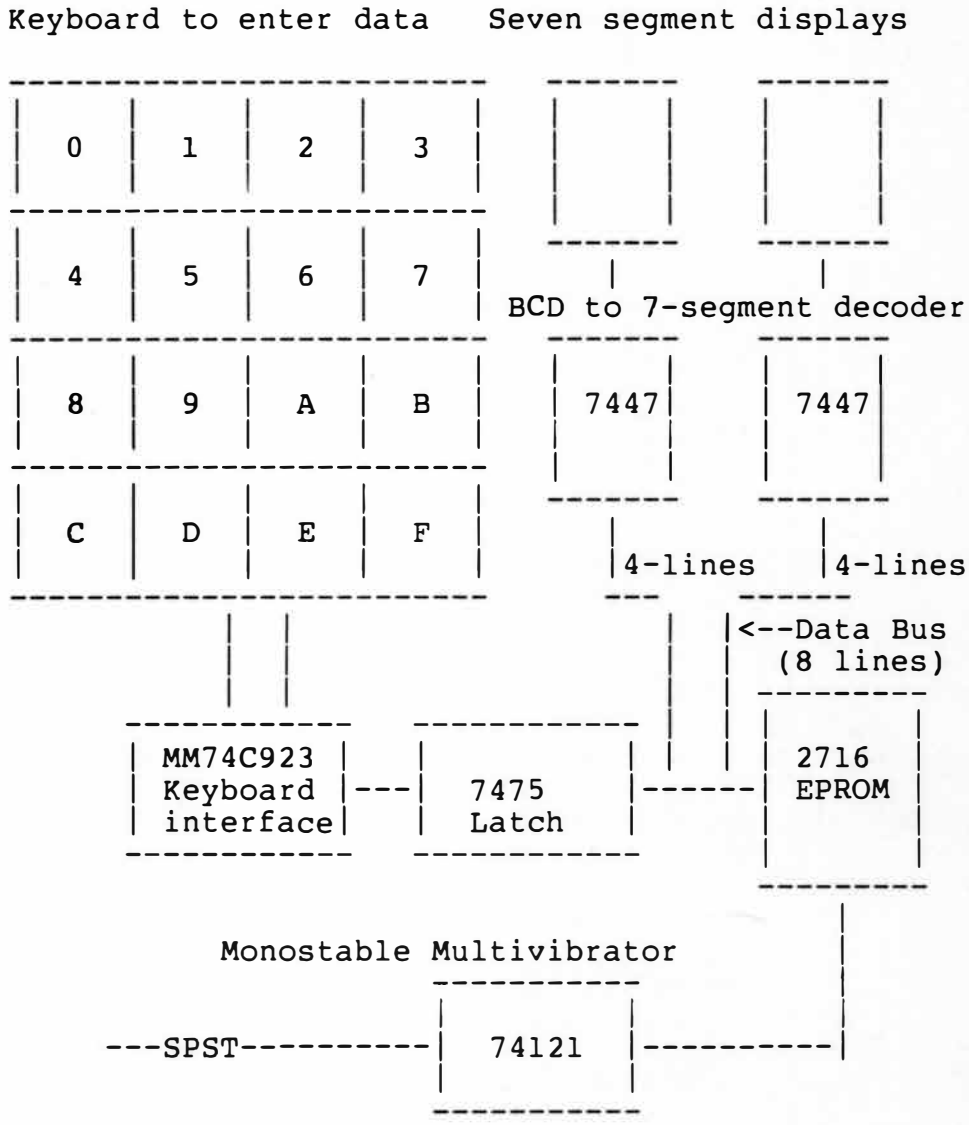


Figure 8. Data Generator

A 20-pin encoder (MM74C923) chip was used to encode the data entered from the keyboard. This is similar to the chip used in a telephone. This chip changes a hexadecimal number to a binary number. This binary number is first latched with a 7475 and then connected to the data bus of the EPROM. The binary number corresponding to the data entry is displayed in the same way as in the address generator using seven segment displays.

Programming Pulse Generator

In most EPROMs and PROMs a programming pulse of specific duration is needed to enter the data into memory. In the case of the 2708 EPROM, this duration is from 1 to 10 ms, while in the case of the 2716 EPROM this duration is 50 to 55 ms. In this experiment the 2716 EPROM was used.

To obtain a specific duration pulse a monostable multivibrator (74121) was used as shown in figure (8). This device generates a single pulse when the SPST switch is closed.

Procedure for using the EPROM programmer:

The EPROM programmer described above can be used for two purposes, reading a ROM (PROM or EPROM) and programming a PROM or an EPROM.

Reading Mode

1. First the EPROM is inserted into its 24-pin socket.

2. The +5V power supply used to power the circuit is turned on and the address bus is reset.

3. The mode switch is set to the read mode.

4. The data and the address (hex) are displayed on the seven segment displays. Data at other addresses can be read by changing the selected address using the SPDT switch.

Programming Mode

1. First insert the proper EPROM into the 24-pin socket.

2. Set the mode switch to the programming mode.

3. Turn the +5V power supply on and then turn the +25V power supply on.

4. Select the desired starting address using SPDT switch and enter the data through the keyboard.

5. A programming pulse of width 50 ms must be sent by the SPST switch to enter the data into the EPROM at the selected address.

6. The address is then changed by SPST switch and the procedure is repeated.

When the entire program has been entered, first turn off +25V power supply and then turn off +5V power supply. The program can then be verified by examining it in the read mode.

CHAPTER V

Z-80 TEST CIRCUITS AND PROGRAMS

We have already discussed the Z-80 CPU in Chapter III. Here we will discuss two simple tests to check the operation of the Z-80 CPU. One program uses only the CPU while the other uses the parallel input/output interface as well. A simple block diagram is given below.

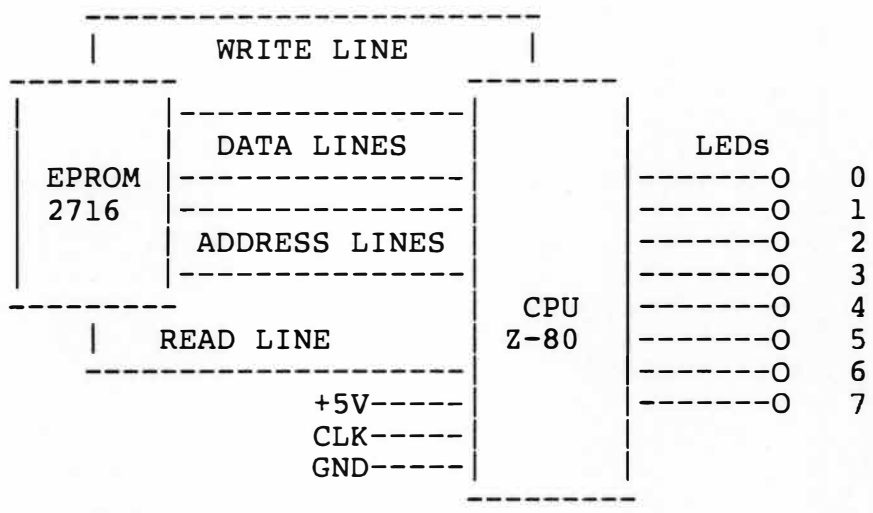


Figure 9. Z-80 CPU and EPROM

The test program is loaded from the EPROM. The EPROM must, of course, be programmed using the EPROM Programmer discussed in Chapter IV. Once the CPU is properly connected (see Appendix D for a detailed drawing) and a five volt square wave TTL (transistor-transistor-logic) signal (1000 Hz) applied to the CPU at the CLK input, the CPU

the CPU will start reading the information from the EPROM.

Test of CPU Operation

A simple program to test the operation of the CPU is given in Appendix B. This program loads the accumulator register with the value 0000 0001 (binary) and the CPU shows this number on the LEDs (light emitting diodes). A left shift (SLA) command is used and so in the next loop the number will be shifted one step left and this time the second LED will light up and the first LED goes out. This process continues until the 8th LED is lit. Then the number is moved one more step left and all LEDs will be off (all bits zero). The whole process will be repeated again and again.

Test of Parallel Input/output Interfacing

In order for the CPU to accept input data from the outside or to transfer data to the outside it is necessary to interface another 40-pin chip called the parallel input/output (PIO). The Z-80 PIO pin configuration is given below. The PIO has two ports, Port A and Port B. All lines of the PIO are connected to the CPU lines of the same name. No address bus is needed for the PIO. Pin number 4 is used for enabling or disabling the PIO. Pin number 5 is used to indicate whether data on the data bus should be treated as data (logic 0) or a control word

(logic 1).

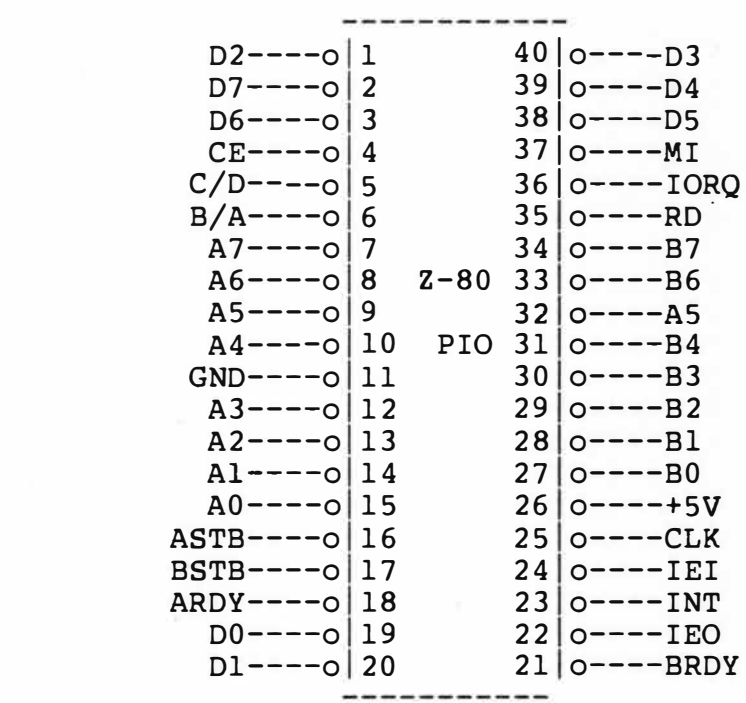


Figure 10. Z-80 PIO

Pin number 6 is used for selecting Port A or Port B of the PIO. Pin numbers 4, 5 and 6 of the PIO are connected to the CPU at addresses A2, A1 and A0 respectively. The PIO can be programmed in four different modes, mode-0 (byte input), mode-1 (byte output), mode-2 (byte input/output for port A only) and mode-3 (bit input/output). Only mode-0 and mode-1 are used here.

A program is needed to configure and use the PIO. A sample program for this purpose is given in Appendix C. This program configures the PIO in such a way that Port A

acts as an input and Port B acts as an output. When input data is present at Port A the program transfers it to the CPU and the CPU then sends this data out through Port B.

The test program must first be stored in the EPROM. After connecting all the circuits properly (as in Figure 9), the power is turned on and the CPU is reset by bringing the reset line momentarily low which loads and executes the program up to the HALT statement (see Appendix C). To continue execution the interrupt line is momentarily brought low. Now the PIO is properly configured and the CPU is ready to receive data through Port A of the PIO.

For example, if an ADC (analog-to-digital converter) is used to generate digital input from an analog signal as shown in Figure 11, the data will be input through Port A. If a DAC (digital-to-analog converter) is used to change the digital information back to analog at Port B the original analog signal will be recovered at the output. The input and output analog signals can be observed and compared on an oscilloscope.

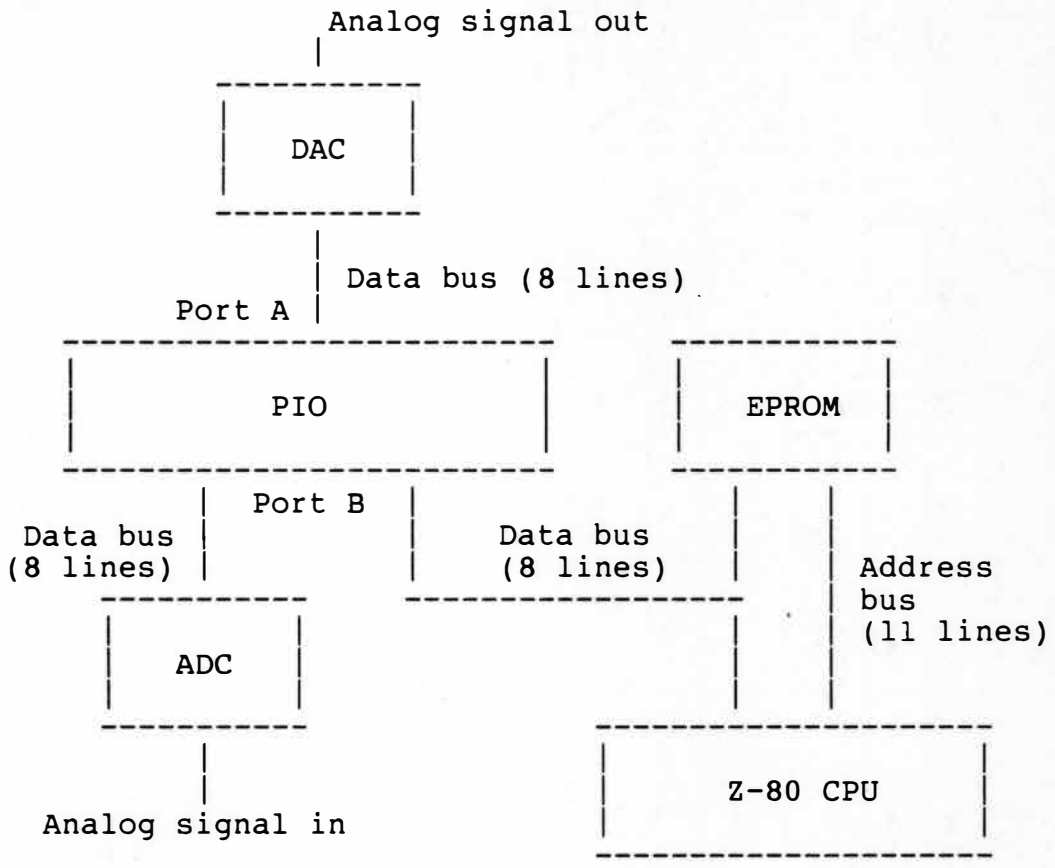


Figure 11. Z-80 CPU and PIO

CHAPTER VI

RAM AND DMAC

In the last chapter we discussed interfacing the EPROM and the PIO to the CPU. In this chapter we will discuss interfacing the RAM and the DMAC to the CPU. As stated before RAM is used for temporary storage. In this work the 2048x8-bit static RAM (MCM6116) chip was used. The following diagram shows the pin assignment for this chip.

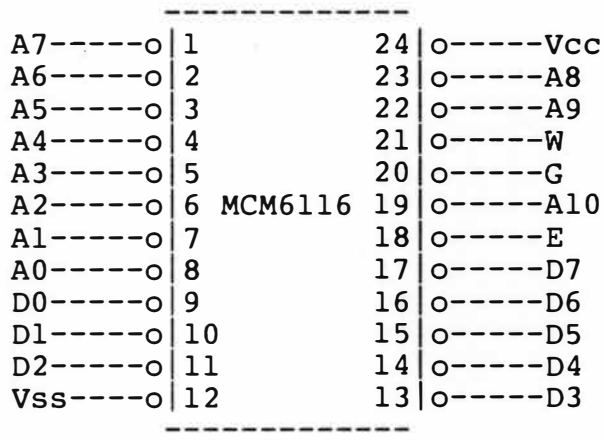


Figure 12. MCM6116 RAM

The pin assignment of the MCM6116 RAM is exactly the same as the MCM2716 EPROM; hence the RAM can be interfaced to the CPU in the same way as the EPROM execpt that pin 18 is connected to address line A11 of the CPU through a NOR gate. To enable the RAM, bit one of this

address line should be high. A block diagram showing the conventional method for storing data in RAM is given below.

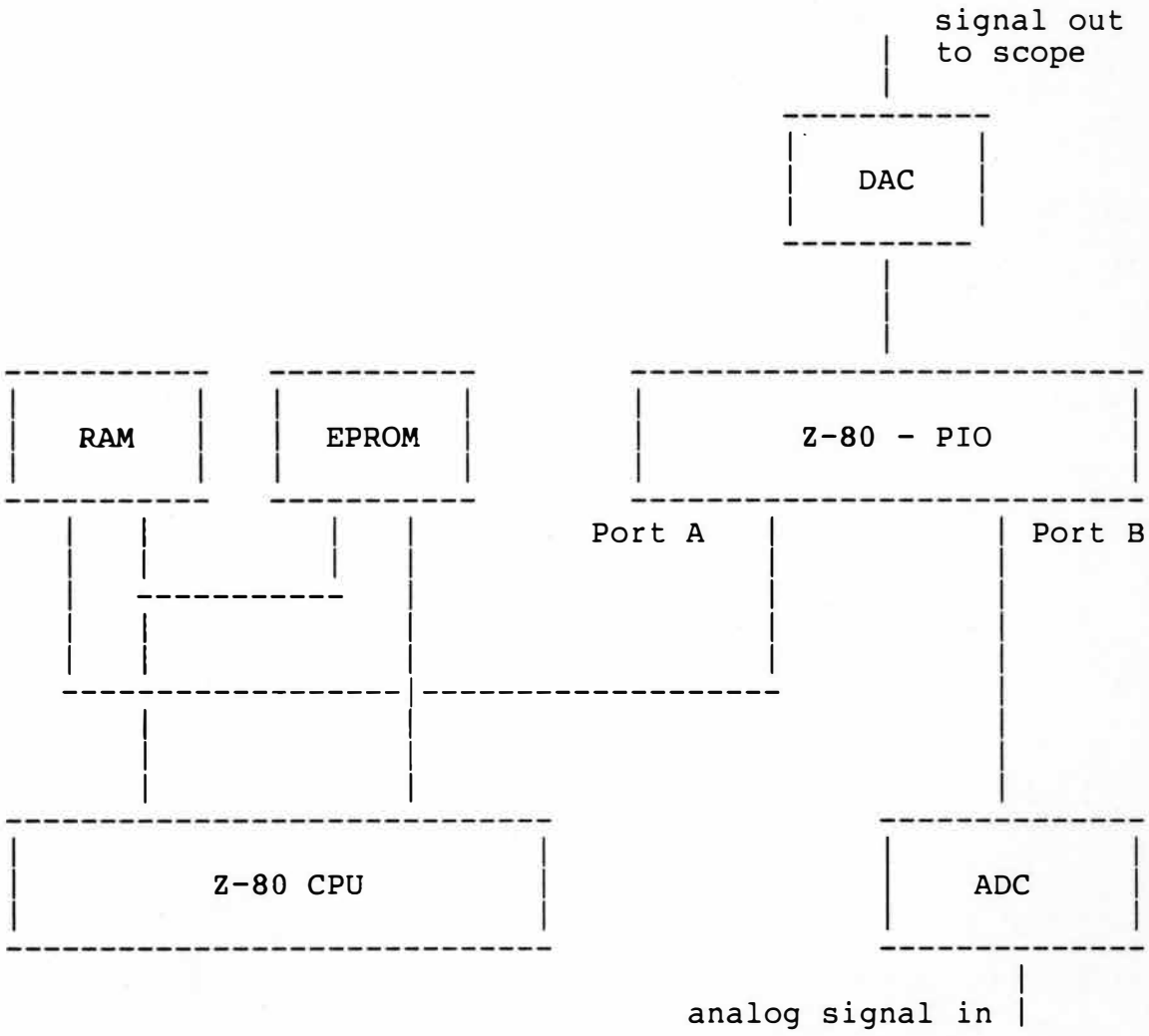


Figure 13. RAM Interfacing

A simple program to store data into RAM is given in Appendix D. This program takes the input data through Port B and stores it in the first 128 memory locations of RAM. Once the data is stored in RAM it can be output through port A and displayed using a DAC, for example.

The data remains in RAM until it is replaced by new data or until the power is turned off.

Z-80 DMAC (Direct Memory Access Control)

The Z-80 DMAC is a powerful and versatile device for controlling and processing the transfer of data. The pin diagram of the DMAC is given in Figure 14. The basic function of the DMAC is to manage the transfer of data between the two ports of the PIO, thereby optimizing transfer speed and control with little external logic.

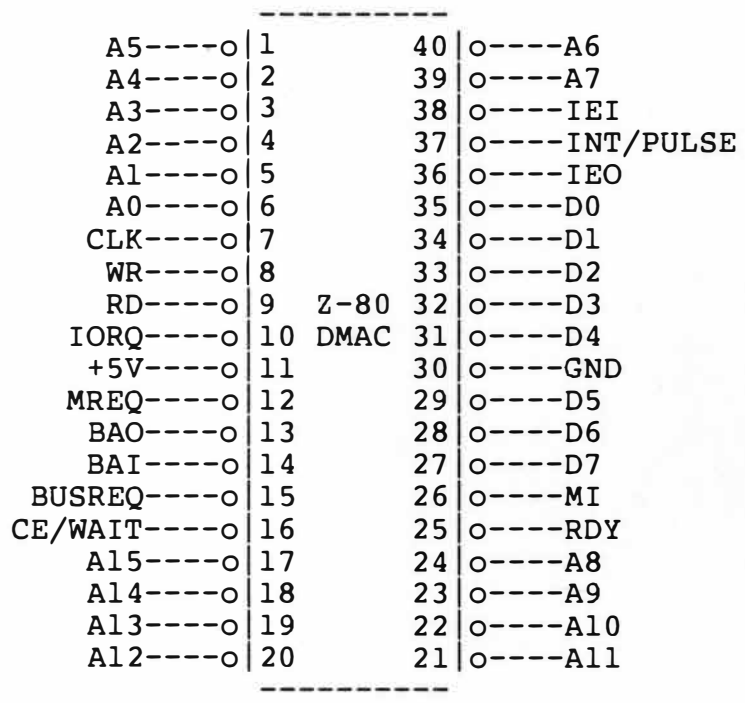


Figure 14. Z-80 DMAC

Transfer can be done from any input (if more than one) to any output. In addition, bit maskable byte searches (only certain bits are masked in a byte) can be performed either

concurrently with transfer or as an operation in itself.

The DMAC can be programmed in three different modes, one byte at a time , burst, and continuous.

One byte at a time

Data operations are performed one byte at a time. Between each byte operation the system buses are released to the CPU.

Burst

Data operations continue until the read line (pin 9) of the DMAC goes inactive. The DMAC then stops and releases the system.

Continuous

Data operations continue until the end of the data is reached before the system buses are released. The end of the data is detected by comparing the next address with address stored in DMAC during its programming.

The CPU must program the DMAC in advance of any data search or transfer by addressing it as an I/O port and sending a sequence of control bytes using output instructions.

Direct Memory Access Interfacing

A block diagram showing how the DMAC is interfaced to the system is shown below.

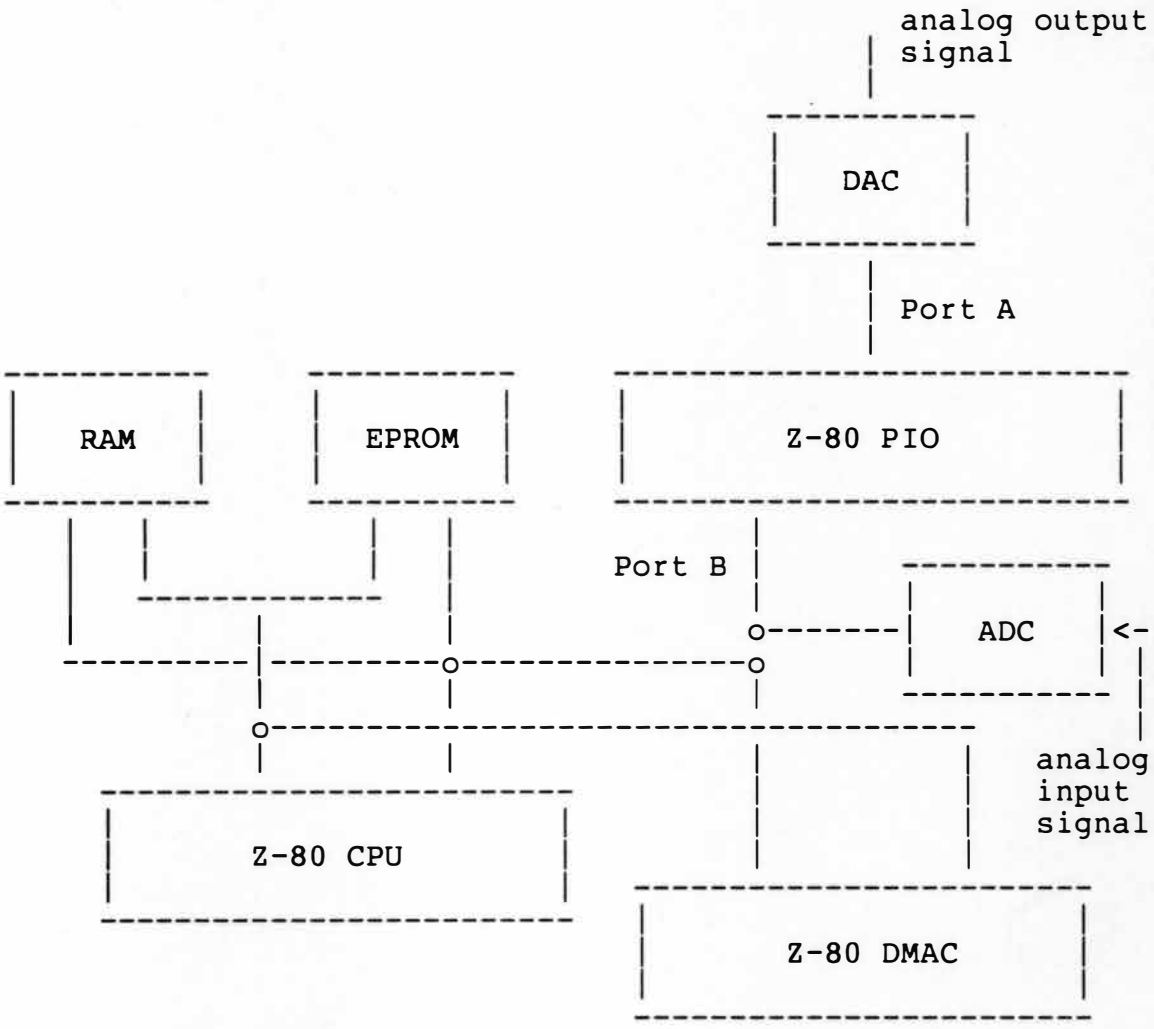


Figure 15. DMAC Interfacing

All the DMAC lines are connected to the CPU pins of the same name. The lines BAI, CE/WAIT, IEI and RDY need some explanation (see Figure 14).

BAI -Bus Acknowledge In (input, active low)

If this line is low then the DMAC is active. Actually BAI (Bus Acknowledge Input) and BAO (Bus Acknowledge Output) make a daisy chain for multiple DMACs. Since only one DMAC is used in the present work the BAI line is connected to ground.

IEI -Interrupt Enable In (input, active high)

This is used with IEO (Interrupt Enable Output) to form a priority daisy chain when there is more then one interrupt driven device. Since just one output device is used here, this line is connected to +5V through a 1K resistor.

CE -Chip Enable (input, active low)

This line is connected to the A2 address line of the CPU through a NOT gate. When this line is low the chip is enabled.

RDY -Ready line (input, programmable, active low)

This line is monitored by the DMAC to determine when a peripheral device associated with the DMAC is ready to read or write.

DMAC programming is quite complex. As an example a small program which will transfer the data from RAM to an *I/O device is given in Appendix E.*

As mentioned before the chip enable (CE) of the DMAC is connected to address A2 of the CPU through a NOT gate, so that when the PIO is enabled the DMAC will be disabled and when the DMAC is enabled the PIO will be disabled. The CPU can program both of these chips separately.

The sample program first configures the PIO and then it stores 2K bytes of data in RAM before the computer executes a halt instruction. The CPU restarts execution when the interrupt line is momentarily brought low. The CPU controls the DMAC in the following way:

1. The system program sets the DMAC to receive the block length of the data to be transferred. Port A has the starting address of the data in memory and Port B is temporarily set as a source.

2. Port A is defined as memory with a fixed incrementing address.

3. Port B is set as a peripheral with a fixed address.

4. The system program sets the DMAC to continuous mode and sets the DMAC to expect the Port B address for input.

5. The DMAC is set to auto reset and RDY is set to high.

6. The Port B address is loaded and the block counter is reset.

7. Port A is set as a source.

8. The DMAC is enabled to start operation.

When the CPU sends the last instruction to the DMAC the CPU is put in a high impedence mode while the DMAC transfers data from RAM to the output device.

The entire DMAC system is now complete and so it is necessary to check its operation very carefully. As a first check the system program given in Appendix D was used with a 1000 Hz clock cycle. This program does not use the DMAC but it takes the data from an input through the PIO and stores it into the RAM. Here the ADC (see Figure 15) is used for conversion of an analog signal to a digital signal. A sine wave signal is converted to digital and stored in RAM. After storing the data the CPU executes a halt statement. Now the input signal is disconnected. By bringing the interrupt line momentarily low the CPU again starts running. The original signal is recovered using a DAC and observed on the oscilloscope. This ensures that every thing is working properly for a 1000 Hz clock cycle.

As a second check the 1000 Hz clock is replaced by a 2.5 MHz crystal clock and the DAC and ADC are replaced by seven segment displays and digital signal switches

respectively. The program is executed and the output is observed at the seven segment displays. This time the same data is stored in all the memory locations of the RAM, e.g., 87(hex). When the stored data is sent to the seven segment displays the same number was observed for each memory location. The reason that this test was conducted without the ADC and the DAC is that these devices were not capable of working at the 2.5 MHz clock cycle.

As a final check the EPROM was replaced with another EPROM with the system program given in Appendix E. This program is used for programming the DMAC. Once the DMAC is programmed it can transfer the data from memory to input/output. Here the digital switches were also used as input and the seven segment displays as output with the 2.5 MHz crystal clock. The same test was repeated, i.e., first the program stored the digital signal and then stopped. The input signal was disconnected and the interrupt line was momentarily brought low thereby giving the output on the seven segment displays. The same output was observed as that which was stored. This result demonstrated that the DMAC was working properly. The *BUSRQ line of the CPU was also checked and found to be high showing that the CPU was in the high impedence state.* The tests were repeated several times each giving the expected results.

The successful completion of this work shows the capabilities of the direct memory acquisition technique. This technique will be used in future fast communication systems like sound processing, digital communication, and data base managment.

APPENDICES

APPENDIX A

TIMING DIAGRAMS FOR THE Z-80 CPU

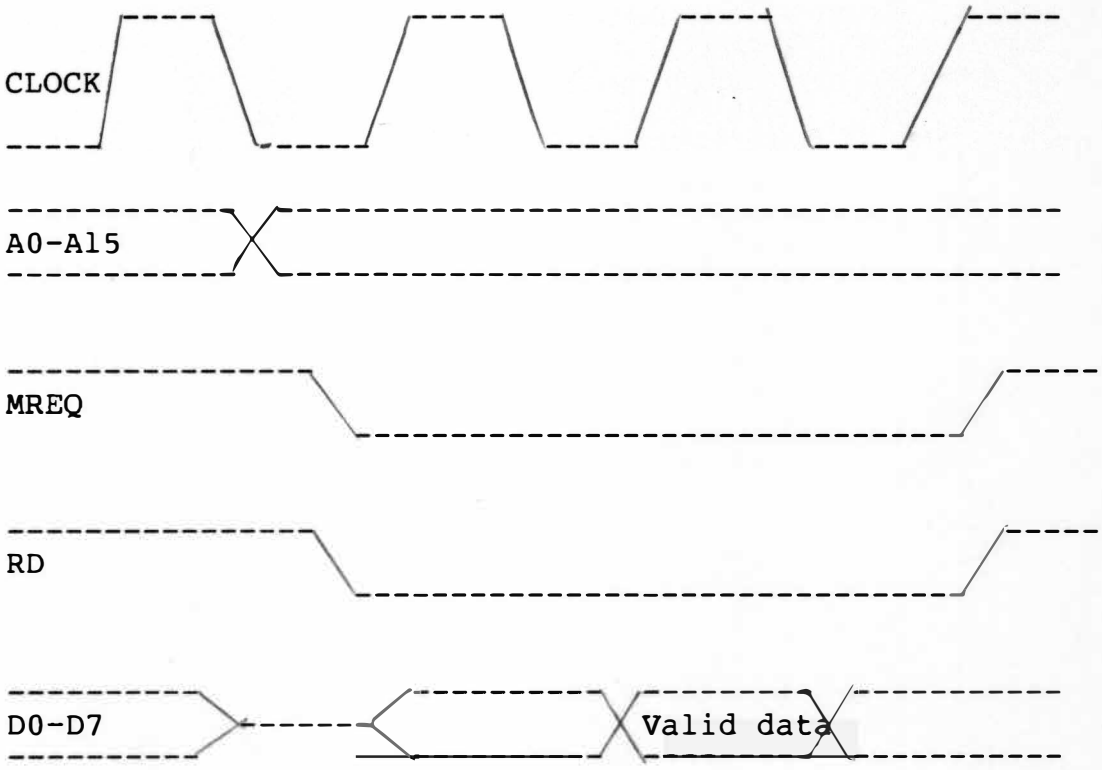


Figure 16. Timing Diagrams

This diagram shows how the address bus, memory request, read line and data bus timing are related to each other during the four cycles of the clock.

APPENDIX B

SYSTEM TEST PROGRAM NO. 1

Before using the EPROM in the circuit shown in figure (9) it must be programmed. A simple program providing a quick check of the CPU operation is given below.

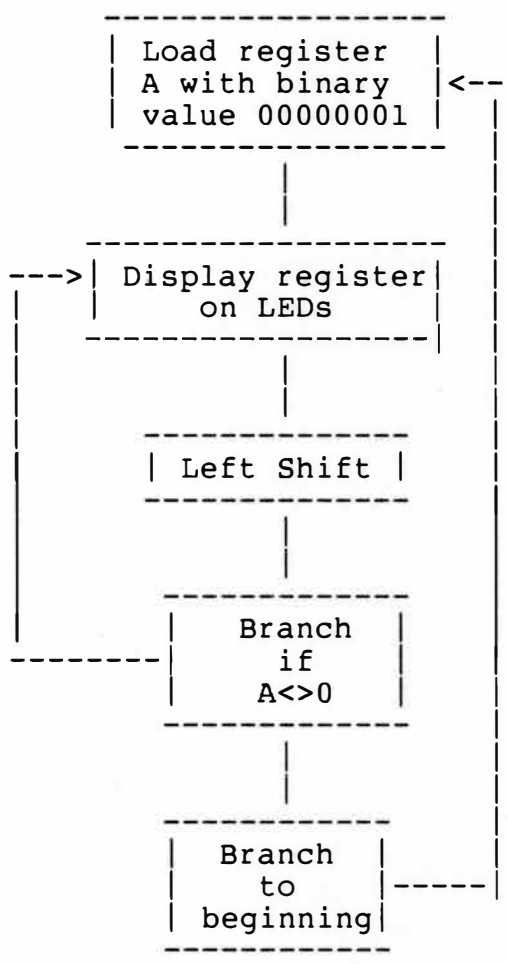


Figure 17. Flow Diagram For System Test Program No. 1

```
0000    3E    LD A, 01          Load 01 in Accumulator A.
0001    01
```

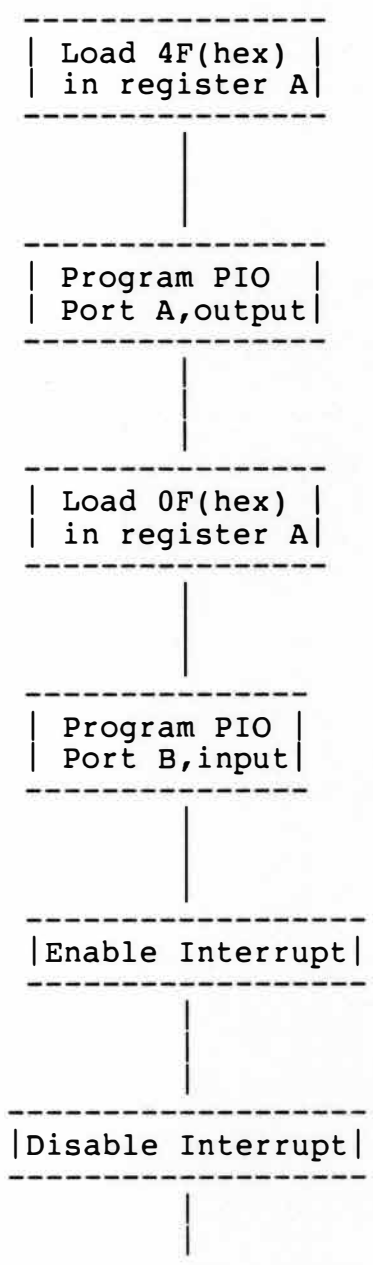
```
0002    D3    Out, 7F          Display contents of A
                               on LEDs
0003    7F
0004    CB    SLA              Shift left
0005    27
0006    C2    JP NZ, 0002      Jump, if not zero to
0007    02                     address 0002
0008    00
0009    C3    JP, 0000         Jump to address 0000
000A    00
000B    00
```

This program will load the accumulator register A with the binary value 0000 0001 and the CPU will display this number on the LEDs. A shift left is executed on the A register and the new value is displayed. This process continues until the logic 1 bit reaches the 8th position. When the next shift left is carried out all bits become zero. The program counter then jumps to address 0000 and the whole process is repeated.

Appendix C

SYSTEM TEST PROGRAM NO. 2

A simple program for configuring the Z-80 PIO is given below.



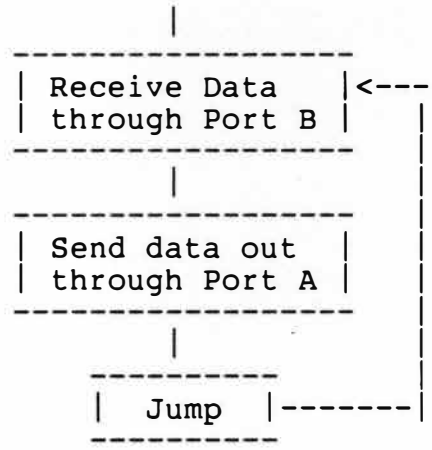


Figure 18. Flow Diagram For System Test Program No. 2

0000	3E	LD A,4F	Load 4F (hex)
0001	4F		
0002	DB	Out(FA), A	Select Port A of
0003	FA		PIO as output
0004	3E	LD A, 0F	Load 0F (hex)
0005	0F		
0006	D3	Out(FB), A	Select Port B of
0007	FB		PIO as input
0008	ED	IM 1	Interrupt mode 1
0009	56		
000A	FB	EI	Enable interrupt
000B	76	Halt	Halt
----	--		
----	--		
0038	F3	DI	Disable interrupt
0039	DB	IN A, (F8)	Receive data through
003A	F8		Port B
003B	D3	Out(F9), A	Send data out
003C	F9		through Port A
003D	C3	JP 0038	Jump to address
003E	38		0038
003F	00		

First the number 4F(hex) is loaded into the accumulator. This number in binary number is

4	F
0100	1111

The 1111(F) indicates that this word is a control word for the PIO. The 0100(4) indicates mode one (output mode) for the PIO.

Out(FA), A specifies accumulator A where FA is the address of the output. FA can be written in binary as

F	A
1111	1010

When the Z-80 CPU executes the Out statment, the lower 8-bits of the address bus carry the address of the I/O device. Pin 6(B/A), pin 5(C/D) and pin 4(CE) are connected to address lines A0, A1, and A2, respectively. The least significant digit of the binary number is zero, so the CPU selects Port A. The second digit is one which means that it is a control word. The third digit is zero which enables the PIO. The rest of the bits do not do any thing. In short this instruction programs the PIO such that Port A will act as an output. Similarly the next two instructions program the PIO such that Port B acts as an input. The instruction at addresses 0008 and 0009 programs the CPU for interrupt mode one (input/output mode). Then a halt instruction is executed bringing the halt line low. The CPU will not execute the next instruction until it is interrupted. This is done by bringing the interrupt line momentarily low. Then the program counter jumps to address 0038 (since the CPU is

programmed in mode one) and starts executing the rest of the program.

After the interrupt is disabled the IN A, (F8) instruction is executed. F8 (hex) can be expanded in binary as

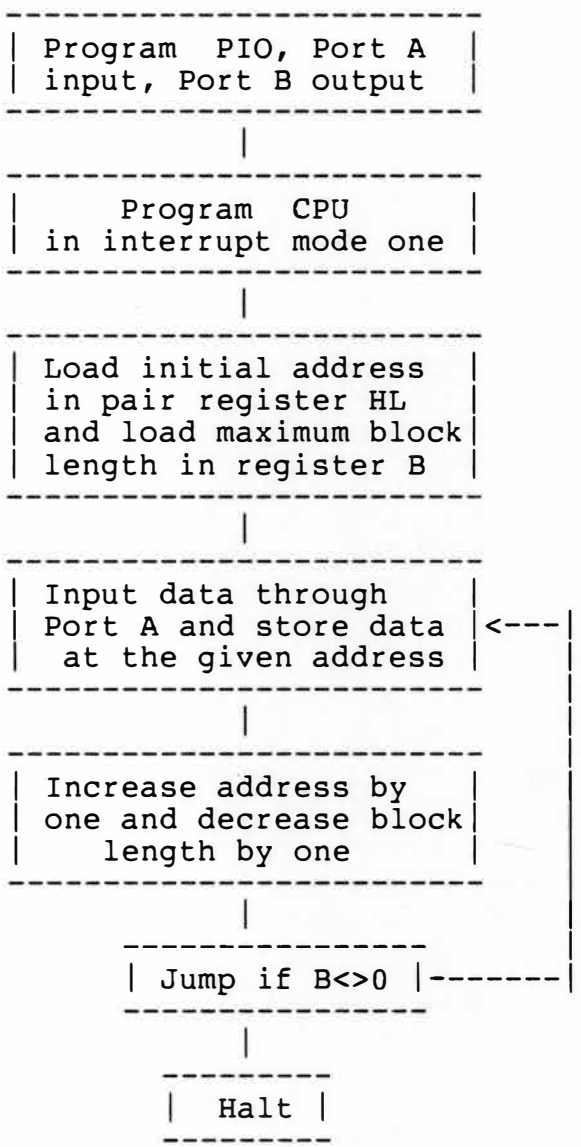
F	8
1111	1000

The first digit (from the right hand side) is zero, so the CPU selects Port A. The second digit is zero and so the information on the data bus is treated as data. The third digit is zero which enables the PIO. At this point the PIO is ready to receive data from the outside.

APPENDIX D

SYSTEM TEST PROGRAM NO. 3

The following is a simple program to enable the system to receive data through port B, store it in RAM, and output the data through port A.



Momemtarily bring the interrupt line low (manualy).

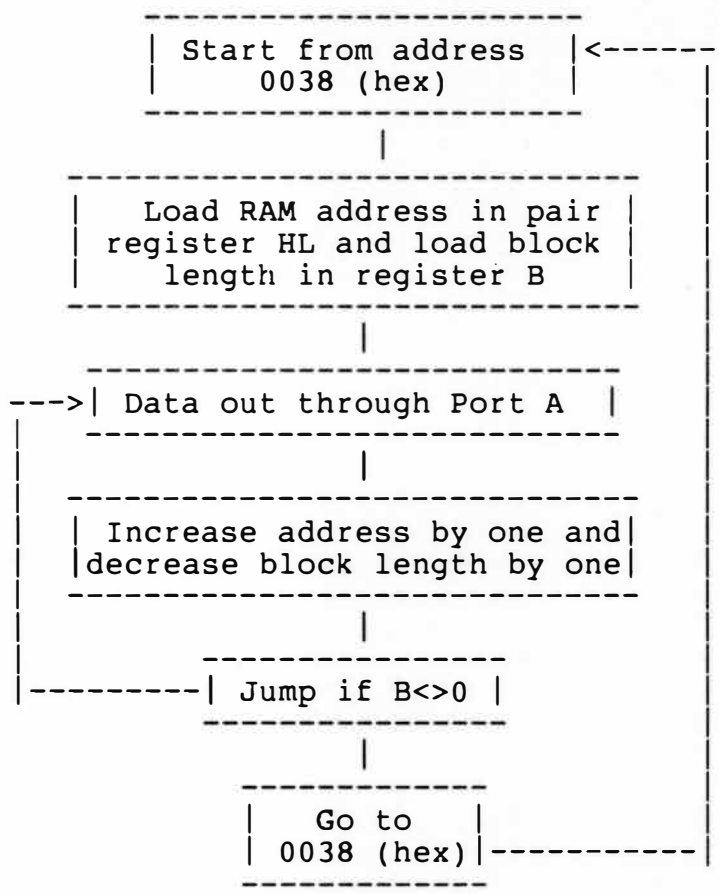


Figure 19. Flow Diagram For System Program No. 3

```
0000    3E   LD A, 4F          Load  4F in register A
0001    4F
0002    D3   Out(FA),A         Select port A as an
0003    FA                     output
0004    3E   LD A, 0F          Load 0F (hex)
0005    0F
0006    D3   Out(FB),A         Select port B as an input
0007    FB
0008    ED   IM 1              CPU interrupt mode one
0009    56
000a    FB   EI                Interrupt enable
000B    F3   DI                Interrupt disable
000C    21   LD HL,F800        Load F800 in pair
000D    00                     register HL
000E    F8
000F    06   LD B,FF           Load FF in register B
0010    FF
```

```
0011    DB    IN A,(F8)          Get input data through
0012    F8                       port B
0013    77    LD(HL),A           Put data at address in HL
0014    78    LD A,B             Load data in A from B
0015    23    INC HL             Increase HL by one
0016    05    DEC B              Decrease B by one
0017    C2    JP NZ,0011         Jump if not zero to 0011
0018    11
0019    00
001A    76    HALT               Halt
----    --
----    --
0038    21    LD HL,F800         Load RAM address in
0039    00                       Pair register HL
003A    F8
003B    06    LD B,FF            Load FF in register B
003C    FF
003D    7E    LD A,(HL)          Load data from (HL)
003E    D3    Out(F9),A          Data out through port A
003F    F9
0040    23    INC HL             Increase data in HL
0041    78    LD A,B             Load A from B
0042    05    DEC B              Decrease data in B by one
0043    C2    JP NZ,003D         Jump if not zero to 003D
0044    3D
0045    00
0046    C3    JP 0038            Jump to address 0038
0047    38
0048    00
```

The first part of this program (from address 0000 to 000B) is the same as in Appendix C. At address 000C the pair register is loaded with F800 which is used for addressing RAM and its memory locations. At address 000F the value FF is stored in register B. Since only 128 or FF(hex) memory location in RAM are used, this number acts as a counter. The instruction IN A,(F8) instructs the CPU to get data from B, where F8 is the address of port B.

In short this program receives the data through port B of the PIO and stores it in RAM from address 0000 to 00FF(hex). When the CPU executes the halt instruction, the next instruction is not executed until interrupt line is momentarily brought low. When this is down the CPU starts execution the next instruction. The remainder of the instructions are used for sending out the data which was stored to port A.

APPENDIX E

SYSTEM TEST PROGRAM NO. 4

The program below directs the Z-80 PIO to receive data through Port B and store these data into RAM. Then the DMAC is programmed to receive data from RAM and send them directly to the DAC.

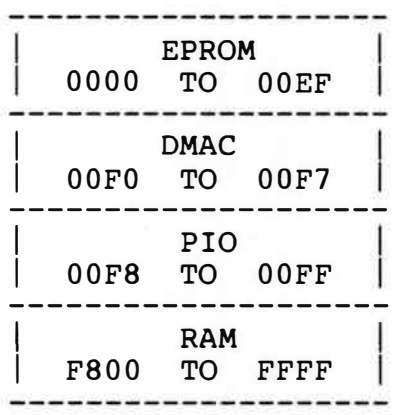
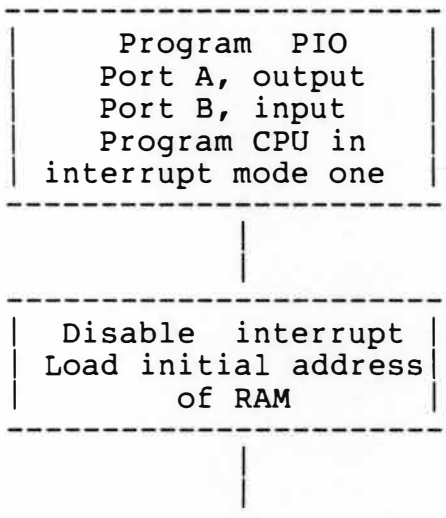
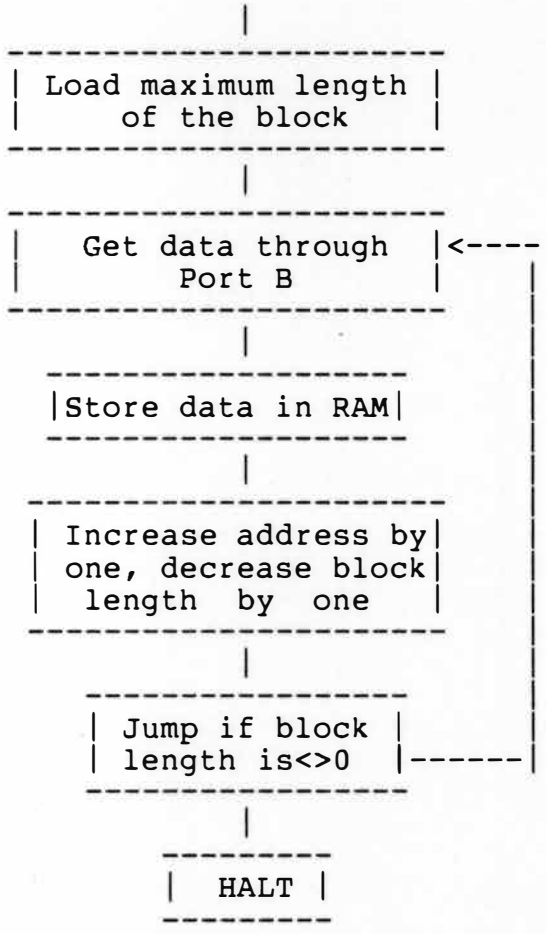
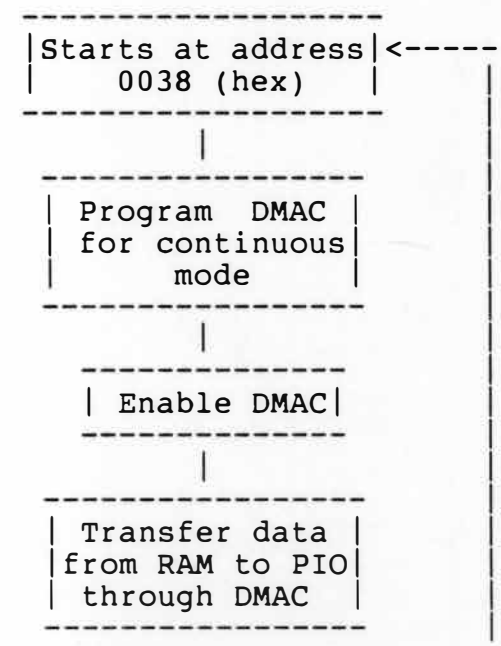


Figure 20. Memory Map





Bring the interrupt line momentarily low (manually).



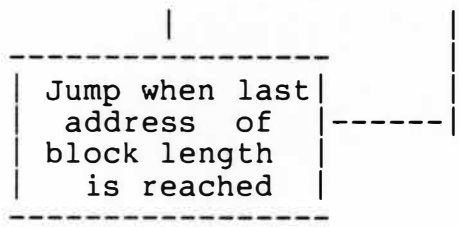


Figure 21. Flow Diagram For Test Program No. 4

```
0000      3E      LD A, 4F      Load 4F in accumulator A
0001      4F
0002      D3      Out(FA),A     Select Port A for output
0003      FA
0004      3E      LD A, 0F      Load 0F in accumulator A
0005      0F
0006      D3      Out(FB),A     Select Port B for input
0007      FB
0008      ED      IM 1          CPU mode one
0009      56
000A      FB      EI            Interrupt enable
000B      F3      DI            Interrupt disable
000C      21      LD HL,F800    Load F800 in pair
000D      00                    register HL
000E      F8
000F      06      LD B,07       Load 07 in register B
0010      07
0011      0E      LD C,FF       Load FF in register C
0012      FF
0013      DB      IN A,(F8)     Get data through
0014      F8                    Port B
0015      77      LD (HL),A     Store data in RAM
0016      23      INC HL        Increase contents of HL
0017      79      LD A,C        Load C from register A
0018      0D      DEC C         Decrease contents of C
0019      C2      JP NZ,0013    Jump if not 0 to 0013
001A      13
001B      00
001C      78      LD A,B        Load A from register B
001D      05      DEC B         Decrease contents of B
001E      C2      JP NZ,0011    Jump if not 0 to 0011
001F      11
0020      00
0021      76      HALT          Halt
----      --
----      --
0038      3E      LD A,79       Load 79 in accumulator A
0039      79
003A      D3      Out(F0),A     Set DMA to  receive
003B      F0                    block length
```

```
003C    3E      LD A,00       Load 00 in accumulator A
003D    00
003E    D3      Out(F0)       Port A lower address
003F    F0
0040    3E      LD A,F8       Load F8 in accumulator A
0041    F8
0042    D3      Out(F0),A     Port A upper address
0043    F0
0044    3E      LD A,00       Load 00 in accumulator A
0045    00
0046    D3      Out(F0),A     Lower bits of block
0047    F0                    length
0048    3E      LD A,08       Load 08 in accumulator A
0049    08
004A    D3      Out(F0),A     Upper bits of block
004B    F0                    length
004C    3E      LD A,14       Load 14 in accumulator A
004D    14
004E    D3      Out(F0),A     Define port A, memory
004F    F0                    with fixed address
0050    3E      LD A,28       Load 28 in accumulator A
0051    28
0052    D3      Out(F0),A     Define port B as I/O
0053    F0                    with a fixed address
0054    3E      LD A,E5       Load E5 in accumulator A
0055    E5
0056    D3      Out(F0),A     Set DMA to continuous
0057    F0                    mode
0058    3E      LD A,05       Load 05 in accumulator A
0059    05
005A    D3      Out(F0),A     Port B lower address
005B    F0
005C    3E      LD A,AA       Load AA in Accumulator A
005D    AA
005E    D3      Out(F0),A     Set DMA to auto reset
005F    F0
0060    3E      LD A,CF       Load CF in accumulator A
0061    CF
0062    D3      Out(F0),A     Load Port B address
0063    F0                    and reset block counter
0064    3E      LD A,05       Load 05 in accumulator A
0065    05
0066    D3      Out(F0)       Set Port A as the source
0067    F0
0068    3E      LD A,CF       Load CF in accumulator A
0069    CF
006A    D3      Out(F0),A     Load Port A address and
006B    F0                    reset block counter
006C    3E      LD A,87       Load 87 in accumulator A
006D    87
006E    D3      Out(F0),A     Enable DMA to start
006F    F0                    operation
```

```
0070        C3      JP 0038      Jump to address 0038
0071        38
0072        00
```

APPENDIX F

CIRCUIT DIAGRAM FOR EPROM PROGRAMMER

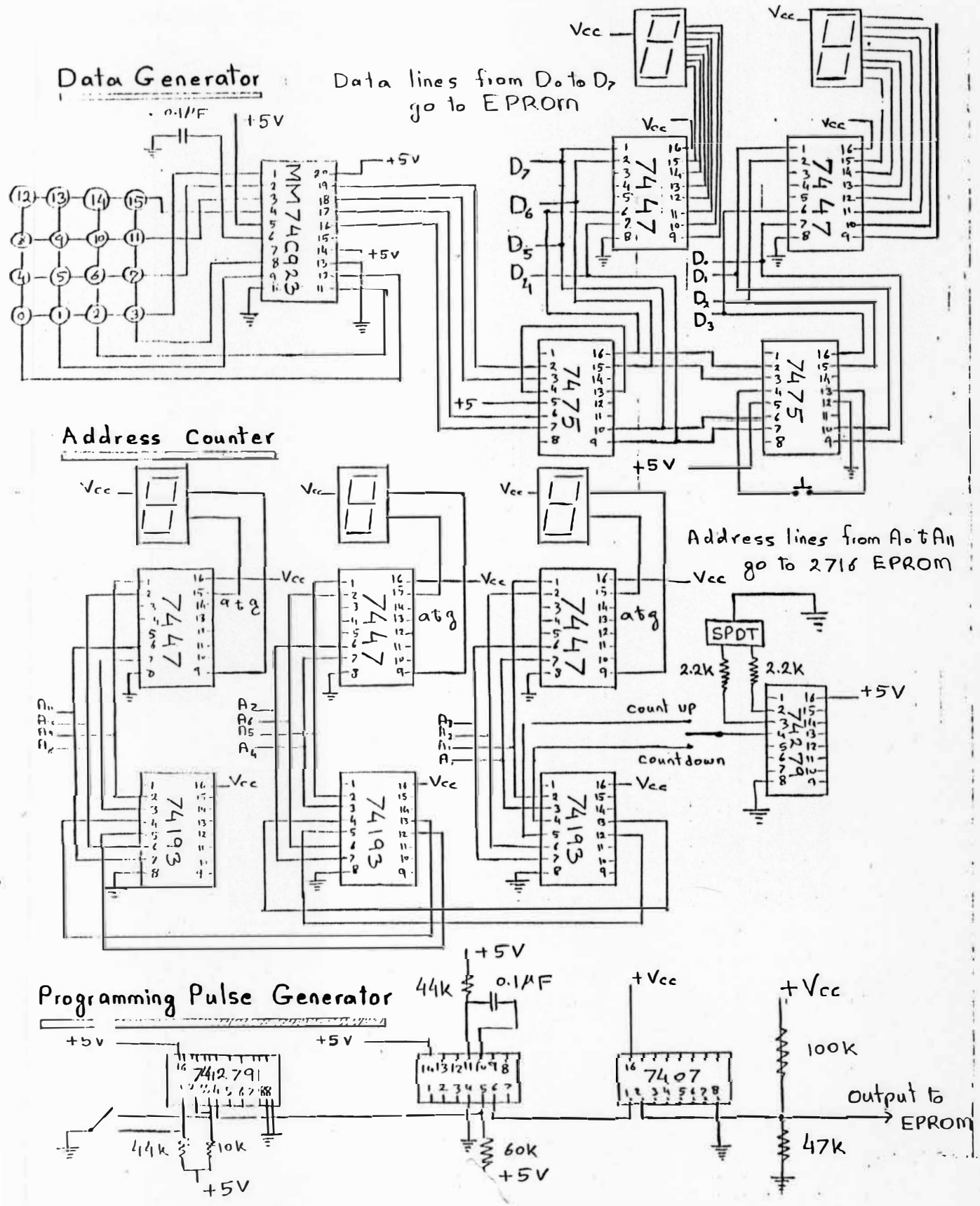


Figure 22. Circuit Diagram For EPROM Programmer

APPENDIX G

CIRCUIT DIAGRAM FOR DMAC

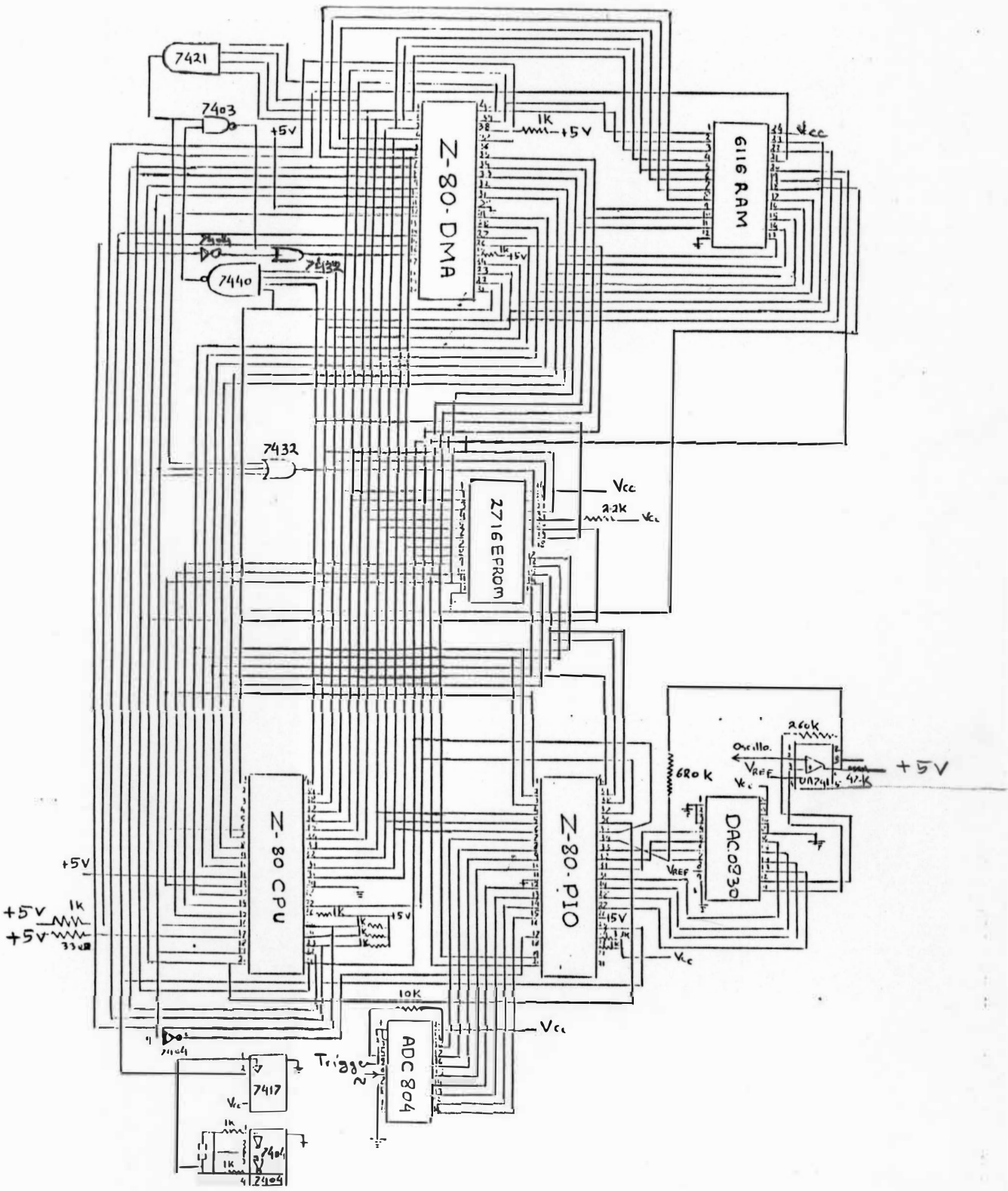


Figure 23. Circuit Diagram for DMAC

BIBLIOGRAPHY

CMOS Motorola Data Book. (1979). Austin, Texas, Motorola Semiconductor Products Inc.

Components Data Book. (1983/1984). Mountain View, California, Zilog Pioneering The Microword.

Joseph, C. Nichols. (1979). Z-80 Microprocessor. Indianapolis, Indiana, E & L Instruments, Inc.

Leventhal, A. L. (1979). Z-80 Assembly Language Programming. Berkeley, California, Howard W. Sams & Co., Inc.

Leventhal, A. L. (1980). Z-8000 Assembly Language Programming, Berkeley, California, Howard W. Sams & Co., Inc.

TTL Data Book for Design Engineers, Second edition. (1981). Dallas, Texas, Texas Instruments Incorporated.