NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

A MICROPROCESSOR INTERFACE FOR THE NM24CF04 SERIAL- ACCESS FERROELECTRIC MEMORY

by

Thomas C. Gonter

December, 1991

Thesis Advisor: Douglas J. Fouts

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (*If applicable*) 32	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000	7b ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (*If applicable*)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (*City, State, and ZIP Code*)	10 SOURCE OF FUNDING NUMBERS Program Element No / Project No / Task No / Work Unit Accession Number

11 TITLE (*Include Security Classification*)
A MICROPROCESSOR INTERFACE FOR THE NM24CF04 SERIAL-ACCESS FERROELECTRIC MEMORY

12 PERSONAL AUTHOR(S) CAPT Thomas C. Gonter

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (*year, month, day*)	15 PAGE COUNT
Master's Thesis	From To	1991 December	102

16 SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (*continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	Ferroelectric memory, FERRAM , NM24CF04, nonvolatile memory

19 ABSTRACT (*continue on reverse if necessary and identify by block number*)

The goal of this study was to demonstrate the feasibility of utilizing ferroelectric memory as a portion of main microprocessor memory. An interface between National Semiconductor's NM24CF04, a nonvolatile, serial-access, ferroelectric memory device, and Intel's 8086 microprocessor was designed and implemented. This thesis discusses the architectural and implementation problems that arise when installing this ferroelectric memory device as a portion of main memory. The actions of the interface are detailed to explain the control and timing requirements necessary to effectively perform write and read operations on the NM24CF04.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS REPORT ☐ DTIC USERS	UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL Douglas J. Fouts	22b TELEPHONE (*Include Area code*) (408) 646-2852	22c OFFICE SYMBOL EC/Fs

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

Approved for public release; distribution is unlimited.

A Microprocessor Interface for the
NM24CF04 Serial-Access Ferroelectric Memory

by

Thomas C. Gonter
Captain, United States Marine Corps
B.S., Texas A&M University, 1985

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1991

ABSTRACT

The goal of this study was to demonstrate the feasibility of utilizing ferroelectric memory as a portion of main microprocessor memory. An interface between National Semiconductor's NM24CF04, a nonvolatile, serial-access, ferroelectric memory device, and Intel's 8086 microprocessor was designed and implemented. This thesis discusses the architectural and implementation problems that arise when installing this ferroelectric memory device as a portion of main memory. The actions of the interface are detailed to explain the control and timing requirements necessary to effectively perform write and read operations on the NM24CF04.

TABLE OF CONTENTS

I. INTRODUCTION 1

A. BACKGROUND 1

B. NATIONAL SEMICONDUCTOR'S NM24CF04 5

1. Pin Descriptions 5

C. 8086 MICROPROCESSOR 7

1. Write Bus Cycle 9

2. Read Bus Cycle 11

II. DATA AND CONTROL PATH DESIGN 13

A. ADDRESS AND DATA BUS STRUCTURE 13

B. DATA FLOW OVERVIEW 15

C. DATA PATH DESIGN 16

D. CONTROL PATH DESIGN 19

1. NM24CF04 Control Bytes 19

2. Main Finite State Machine 21

3. Counter 26

III. IMPLEMENTATION 27

A. NM24CF04 SDA BUS OPERATION 27

B. FINITE STATE MACHINE IMPLEMENTATION 30

C. MEMORY CHIP SELECTION 34

D. DATA PATH IMPLEMENTATION 35

E. WRITE CYCLE IMPLEMENTATION 38
F. READ CYCLE IMPLEMENTATION 42

IV. TESTING PROCEDURE 46
A. TESTING STRATEGY 46
1. Control Circuitry 47
2. Ferroelectric Memory 48

V. CONCLUSIONS 55

APPENDIX A 58
A. PROGRAM 'FSM1.PRG' 58
B. FILE 'FSM1.INFO' 60
C. PROGRAM 'FSM2.PRG' 63
D. FILE 'FSM2.INFO' 63

APPENDIX B 64
A. PROGRAM 'EPLD1.ABL' 64
B. PROGRAM 'EPLD2.ABL' 67
C. PROGRAM 'EPLD3.ABL' 70
D. PROGRAM 'EPLD4.ABL' 71
E. PROGRAM 'EPLD5.ABL' 74
F. PROGRAM 'EPLD6.ABL' 77
G. PROGRAM 'EPLD7.ABL' 78
H. PROGRAM 'EPLD8.ABL' 79
I. PROGRAM 'EPLD9.ABL' 81

J. PROGRAM 'EPLD10.ABL' 83

APPENDIX C 85

A. BOARD LAYOUT 85

B. SCHEMATICS 86

LIST OF REFERENCES 91

BIBLIOGRAPHY 92

INITIAL DISTRIBUTION LIST 93

LIST OF FIGURES

Figure 1.1 Ferroelectric Capacitor, [Ref.1]. 2

Figure 1.2 Logical Representation Through Current Flow, [Ref.2]. 3

Figure 1.3 NM24CF04 Pin Diagram. 6

Figure 1.4 Typical BIU Bus Cycle, [Ref. 3]. 9

Figure 1.5 8086 Write Bus Cycle, [Ref. 3]. 10

Figure 1.6 Wait State Insertion, [Ref. 4]. 11

Figure 1.7 8086 Read Bus Cycle, [Ref. 3]. 12

Figure 2.1 System Bus Structure. 14

Figure 2.2 Interface Block Diagram. 15

Figure 2.3 Data Path. 17

Figure 2.4 Byte Formats. 21

Figure 2.5 Main Finite State Machine. 22

Figure 2.6 Counter Finite State Machine. 26

Figure 3.1 Definition of START and STOP, [Ref. 5]. . . 28

Figure 3.2 Acknowledge Response from Receiver, [Ref. 5]. 29

Figure 4.1 Ferroelectric Memory Write Cycle. 51

Figure 4.2 Ferroelectric Memory Read Cycle. 52

I. INTRODUCTION

A. BACKGROUND

Semiconductor memories, such as dynamic and static random access memories (DRAMSs and SRAMs), currently dominate the fast access memory areas of modern computer systems. However, they have one critical deficiency, they lose their contents when power is interrupted. Due to recent technological breakthroughs, ferroelectric memory is now being offered as an alternative, nonvolatile data storage device.

The concept of a ferroelectric memory is by no means a new discovery and the term *ferroelectric* is deceptive since no iron is used. Ferroelectric's only link with ferromagnetics is the fact that the Q-V hysteresis loop of a ferroelectric device is similar to a B-H loop in ferromagnetics. During the 1950's and 1960's several researchers investigated ferroelectrics as a replacement for iron core memory, [Ref. 1]. However, their efforts were abandoned because of technological issues such as structural fatigue (endurance), inability to produce high quality submicron films, high voltage switching thresholds, and addressing by row and column lines could not be accomplished. Breakthroughs in thin-film and CMOS technologies are responsible for ferroelectric's resurgence as a viable source of nonvolatile data storage.

One method of building a ferroelectric memory element is known as the ferroelectric capacitor. This device, shown in Figure 1.1, consists of a layer of ferroelectric material, such as lead zirconate titanate (PZT), serving as the dielectric between two electrodes. When the element is being written to (charged), the bipolar molecules are aligned in one direction (+ on one end and - on the other). The element is read by applying another voltage across the capacitor and observing the current pulse generated by the charging current of the capacitor.

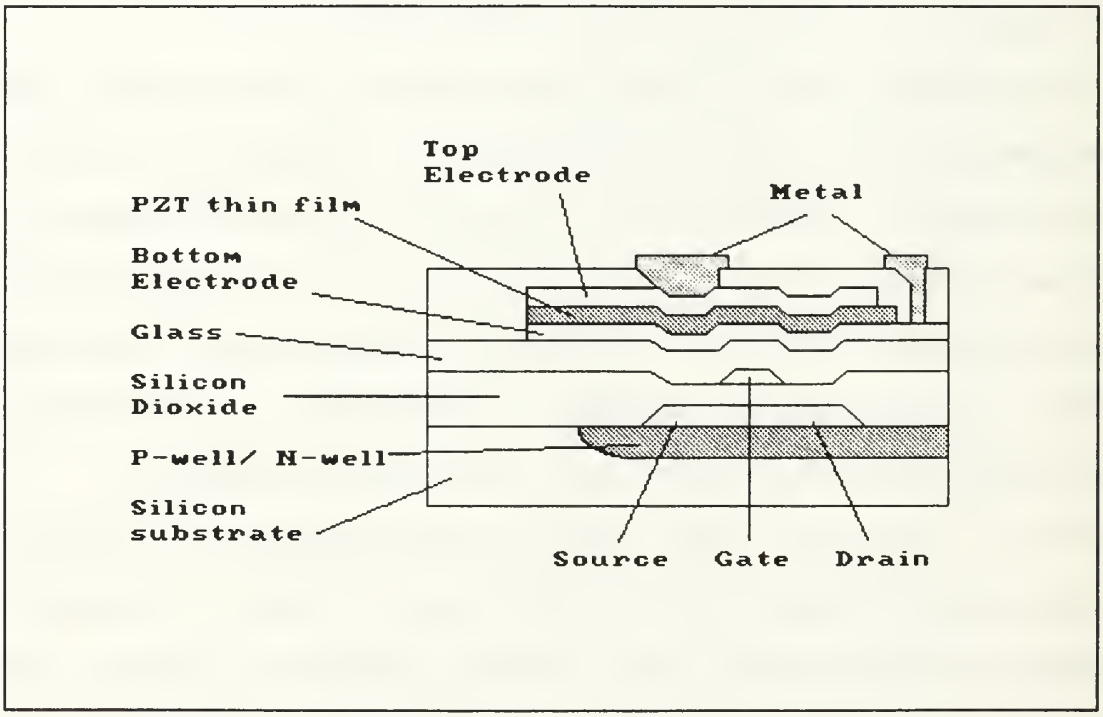


Figure 1.1 Ferroelectric Capacitor, [Ref.1].

A small current pulse indicates the proper alignment of molecules. However, a large current pulse indicates that the molecules had to relocate to opposite ends of the device in

response to the applied voltage. Therefore, a logic '0' or '1' can be determined by the switching current present whenever the sampling reference has been applied.

In Figure 1.2, it is assumed that a '0' is stored in the device. With a voltage applied to the capacitor, the molecules are in the correct position with respect to the applied voltage, resulting in minimal current flow.

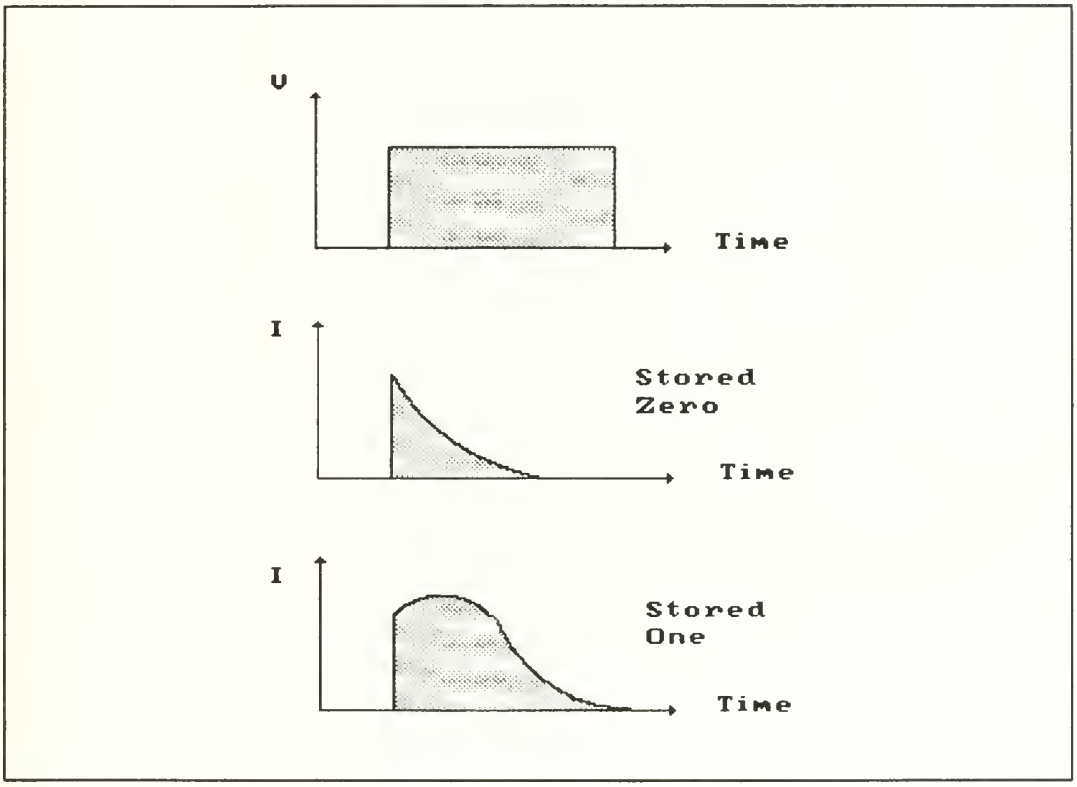


Figure 1.2 Logical Representation Through Current Flow, [Ref.2].

However, if a '1' was stored in the capacitor, the molecules would be aligned in the opposite direction, and more current flows. Notice that this is a 'destructive read' and the information must be written back to the device after every read operation. Another important aspect of ferroelectric

thin-films are their high degree of radiation hardness.

For PZT, the radiation hardness exceeds 5Mrad total dose and 2×10^{11} rad/s. Therefore, ferroelectric memories are attractive for satellite use where the device will be exposed to high radiation. Semiconductor manufacturers are beginning to mass produce ferroelectric memories utilizing integrated circuit technology.

Engineering samples of National Semiconductor's NM24CF04 were obtained and this thesis demonstrates the feasibility of installing the NM24CF04 as a portion of main memory. The speed performance of the NM24CF04 is also addressed due to the serial nature of the device. Hardware was designed and implemented to overcome architectural problems so that the device was incorporated as a functional block of addressable 8086 memory.

The contents of this thesis include:

Chapter I: Introduces the ferroelectric technology, NM24CF04, and 8086 microprocessor.

Chapter II: Contains a discussion of the architectural and timing considerations that were necessary to develop the design of the interface between the NM24CF04 and 8086 microprocessor.

Chapter III: Describes the implementation of the final circuit. Lessons learned from the implementation are also discussed.

Chapter IV: Discusses the strategy employed to test the final circuit and the results of the testing.

Chapter V: Summarizes the study and includes recommendations for areas of further research.

B. NATIONAL SEMICONDUCTOR'S NM24CF04

The NM24CF04 is fabricated with CMOS ferroelectric technology and provides 512 bytes of ferroelectric memory cells. The chip is internally organized as two 256 byte pages. The NM24CF04 is compatible with Xicor's X24C04 EEPROM. However, National Semiconductor's ferroelectric version provides 10^{11} write cycles where the Xicor product only allows 10^{5} write cycles. The improvement in the number of write cycles makes it an attractive prospect to become a functional block of main memory, providing the nonvolatility of ROM and the flexibility of RAM. The chip uses a two wire serial interface and a bidirectional data transfer protocol to move data between itself and the requesting device.

1. Pin Descriptions

Figure 1.3 shows the pin diagram for the NM24CF04.

Serial Clock Line (SCL): The SCL pin is used to clock data into and out of the device. The NM24CF04 is capable of operating at a maximum clock frequency of 100 kHz. The clock is provided by the external master device requiring service and should be removed when the chip is not being addressed. The SCL is an open drain input requiring a pull-up resistor.

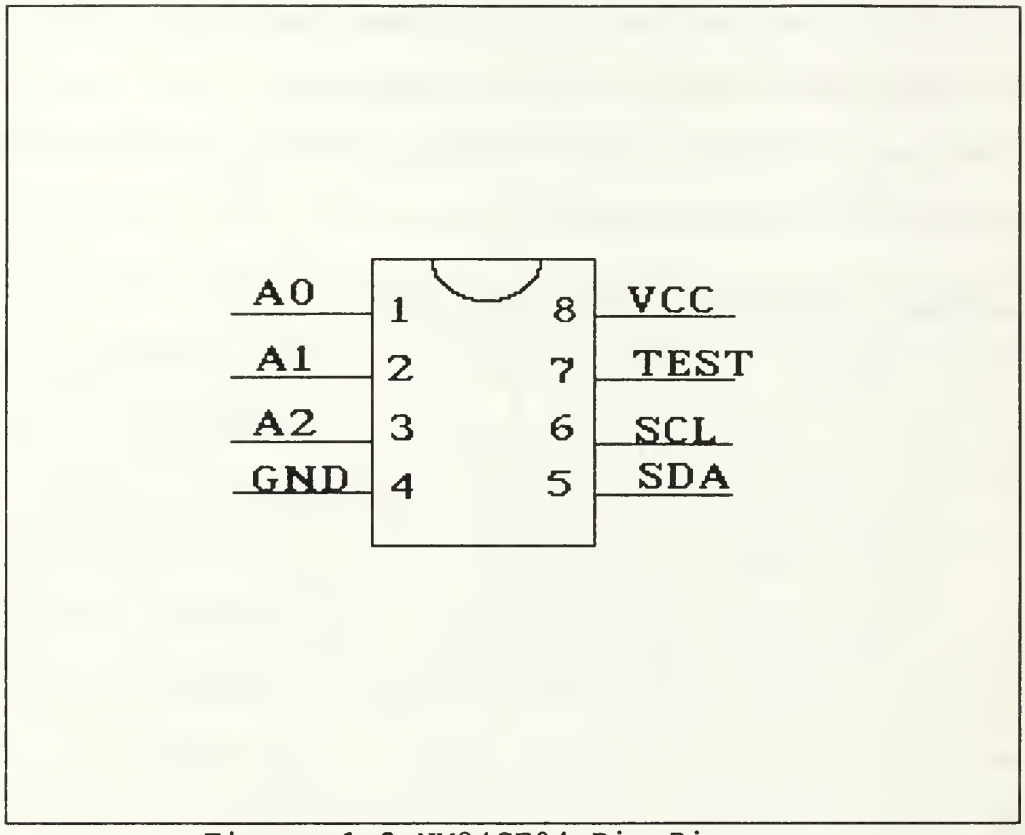


Figure 1.3 NM24CF04 Pin Diagram.

Serial Data Line (SDA): The SDA pin is bidirectional and allows data transfer into and out of the chip. It is also an open drain output and requires a pull-up resistor. The open drain output may also be wire ORed with other open drain devices. This is how four devices are connected on the same SDA bus.

TEST: This pin is unused, but must be wired to ground.

A0: This pin is unused by the NM24CF04. However it must be wired to ground for proper device operation.

A1,A2: These two pins form the LSB's of the six bit slave address. The four MSB's of the slave address are always represented by the binary pattern '1010'. Whenever access of a ferroelectric memory chip is desired, the requesting device must get the attention of the proper device by transmitting the slave address of the desired chip onto the SDA bus. Four separate NM24CF04's may reside on the same SDA bus and the binary combinations of pins A1 and A2 provide the ability to address four different devices. The pins may be driven or static. The design employed the static approach and connected the pins to either +5 volts or ground to establish the hardwired address for each device. Table 1.1 shows the possible combinations.

TABLE 1.1 Bank Selection.

A2	A1	BANK #
GND	GND	0
GND	V_{cc}	1
V_{cc}	GND	2
V_{cc}	V_{cc}	3

C. 8086 MICROPROCESSOR

The 8086 microprocessor's standard operating speed is 5 MHz. The microprocessor operates on an 8-bit or 16-bit multiplexed bus. The device can address up to 1 megabyte of memory and operate in one of two modes of operation. In minimum mode the CPU emits the bus control signals for memory,

where in maximum mode an 8288 Bus Controller assumes the responsibility of controlling all devices on the system bus. The described design utilizes the 8086 configured in minimum mode. The CPU of the 8086 is further divided into two separate processing units. The execution unit (EU) executes instructions and the bus interface unit (BIU) fetches instructions, reads operands, and writes results.

While the processor is busy executing instructions the BIU looks ahead and fetches additional instructions from memory. The retrieved instructions are stored in the instruction stream queue. The 8086's instruction stream queue can store up to six instruction bytes. The BIU normally fetches two bytes unless the fetch is from an odd address, where the BIU only fetches the byte from the odd address. The BIU is responsible for executing all external bus cycles to read data from memory or write data to memory. All bus cycles consist of a minimum of four clock cycles or "T-states" known as T_1, T_2, T_3, and T_4. Figure 1.4 shows a typical BIU bus cycle.

A bus cycle consists of a chain of events in which the address of memory location is output, then a read or write control signal is presented, followed by the data in a write operation. The addressed device accepts the data on a write cycle or places data on the bus during a read operation.

For slow devices, such as the NM24CF04, wait states must be inserted between T_3 and T_4. During a wait state the data on the bus remains unchanged. Upon completion of the cycle, memory latches data written or the BIU removes data read.

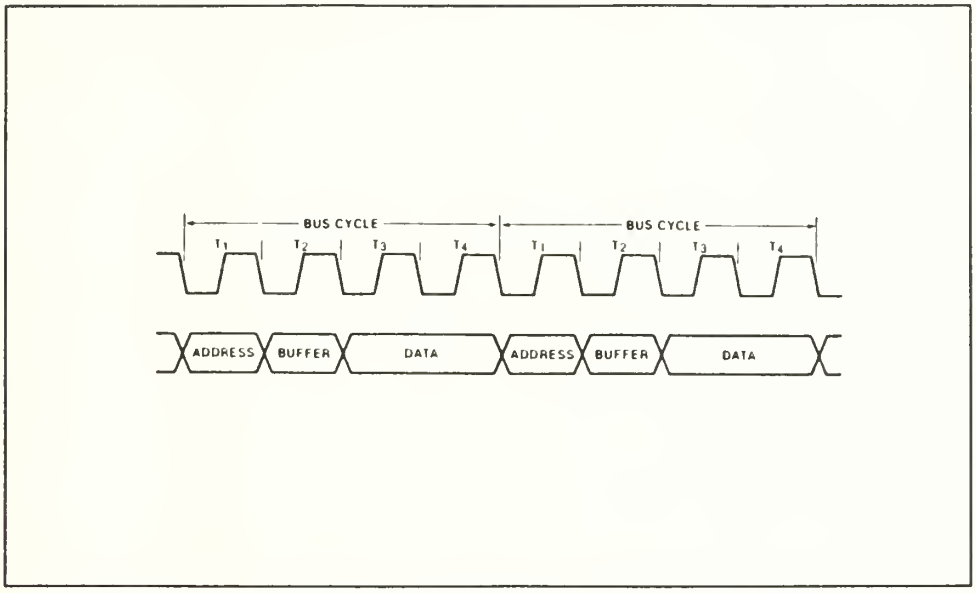


Figure 1.4 Typical BIU Bus Cycle, [Ref. 3].

1. Write Bus Cycle

Figure 1.5 shows the write bus cycle. The CPU places the write address on the multiplexed address/data bus during T_1. During state, T_1 Address Latch Enable (*ALE*) is asserted high to latch the write address into banks of 74LS373 octal latches. The state of these latches can then be decoded by combinational logic to generate the necessary chip select for memory. Upon completion of T_1, the CPU places the write data on the multiplexed bus (*AD15* - *AD0*) from T_2 until T_4.

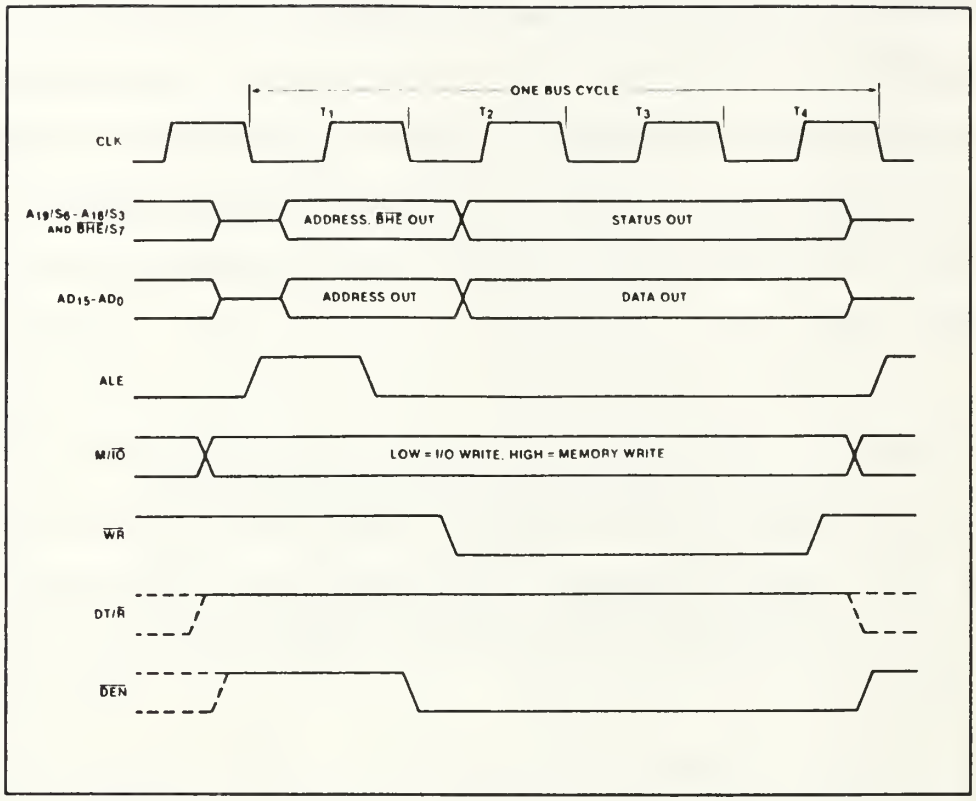


Figure 1.5 8086 Write Bus Cycle, [Ref. 3].

During state T_2, $\overline{WR}$ is asserted low to indicate to memory that a write operation is requested. The *READY* signal is used to force the CPU to insert wait states in the bus cycle. To insert a wait state, *READY* must be low prior to the end of state T_2. Figure 1.6 shows the timing relationship required to generate a wait state. Signal *READY* is produced by an 8284 Clock Generator and Driver, which generates the signal depending on combinational logic inputs. It is the responsibility of the user when addressing memory or I/O peripherals to realize that if data transfers require more time than is allowed via the standard bus cycle that *READY* be asserted low to insert wait states. The user must also assert

READY high after the proper data transfer time has passed so that the bus cycle may terminate.

Signal Data Transmit/Receive ($DT/\overline{R}$) controls the flow of data through banks of 8286 octal bus transceivers when operating in the minimum mode configuration. With the signal high, data can flow from the 8086 microprocessor onto the data bus and into the appropriate memory location. Data Enable ($\overline{DEN}$) is provided as an output enable for the data transceivers.

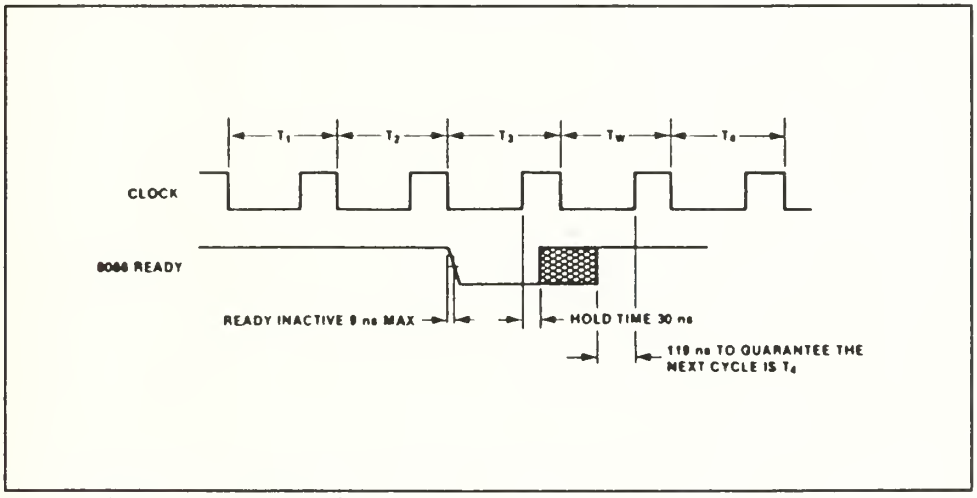


Figure 1.6 Wait State Insertion, [Ref. 4].

2. Read Bus Cycle

Figure 1.7 shows the 8086 Read Bus Cycle. The cycle is very similar to the write cycle and requires four "T" states. The CPU places the read address on the multiplexed address/data bus during T_1. Again, during state T_1, Address Latch Enable (*ALE*) is asserted high to latch the write address into banks of 74LS373 octal latches. These latches are

decoded to generate the necessary chip select for memory. Upon completion of T_1, the CPU floats the address/data bus in preparation for data to be received from memory. During state T_2, $\overline{RD}$ is asserted low to indicate to memory that a read operation is requested. Again, if reading from a slow memory device, the user is responsible for adding logic to control *READY* and generate wait states prior to state T_3. Data is expected and sampled on the data bus at state T_3. Signal $DT/\overline{R}$ is asserted low to configure transceivers in the receive mode. Signal $\overline{DEN}$ is asserted low to enable the transceivers to pass the received data to the CPU.

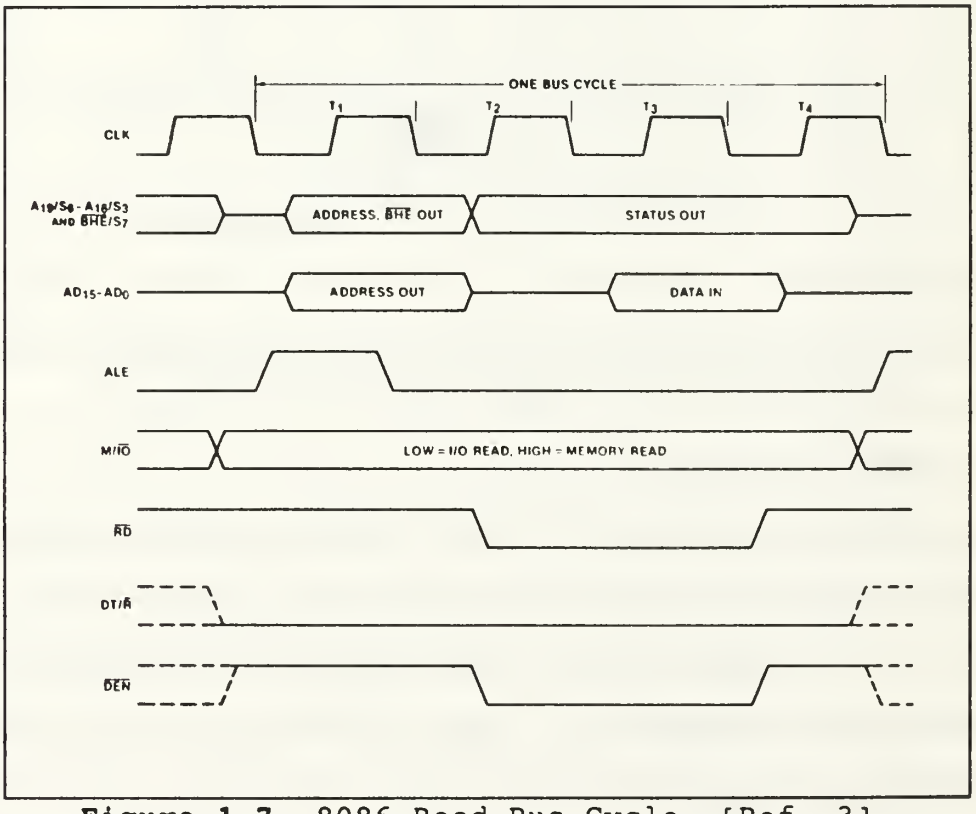


Figure 1.7 8086 Read Bus Cycle, [Ref. 3].

II. DATA AND CONTROL PATH DESIGN

A. ADDRESS AND DATA BUS STRUCTURE

Figure 2.1 shows the system bus structure implemented. The 8086 has the ability to address one megabyte of memory. It is also capable of 8 bit or 16 bit data transfers. The 8086 has a 20 bit address bus, A19-A0. The address space is split into two banks of 512K bytes each. One bank connects the lower half of the data bus (D7-D0) and corresponds to even addresses (A0 = 0). The remaining bank connects the the upper half of the data bus (D15-D8) and corresponds to odd addresses (A0 = 1). Address lines A19-A1 specify a particular byte in each bank. To perform a byte transfer to an even address the 8086 specifies an address with A0 low and signal Bus High Enable (*BHE*) high. With *BHE* high the upper bank of memory or odd byte of the word address is disabled from and memory access.

To perform a byte transfer to an odd address the 8086 asserts BHE low and A0 high. This allows access to the upper bank while disabling any memory access of the lower bank or even byte. To perform a word transfer of 16 bits the 8086 asserts both BHE and A0 low. This enables both memory banks and allows transfer of the entire data bus D15-D0 during one bus cycle.

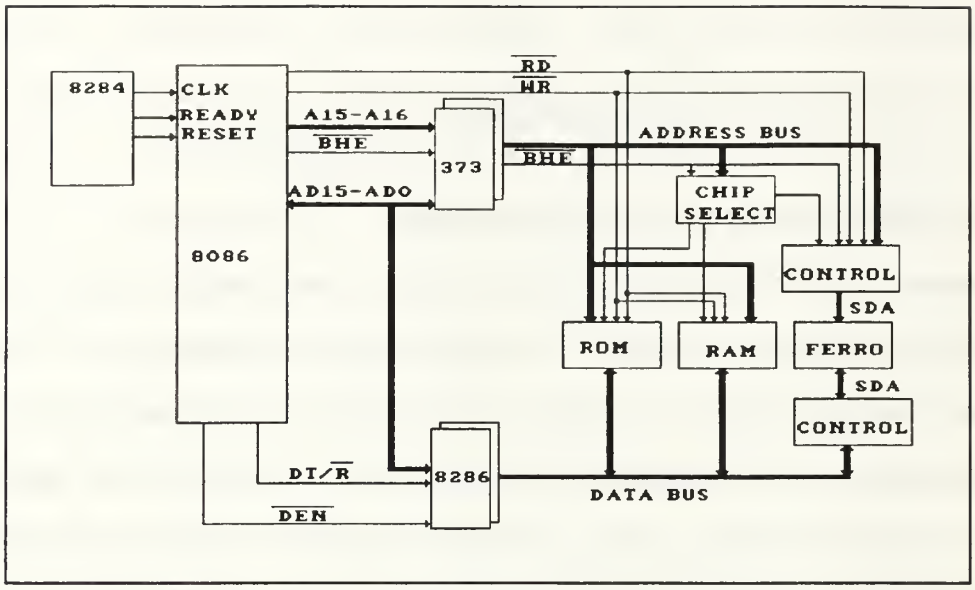


Figure 2.1 System Bus Structure.

The design effort did not require a system with an address space of 1 megabyte. The design required 4K bytes of ferroelectric RAM, 8K of ROM, and 8K of static RAM. The memory selected is dependent on A14, A13, and A12. Table 2.1 shows the memory selected based upon the address on the address bus.

Table 2.1 Memory Selection.

ADDRESS LINES A14,A13,A12	MEMORY ACCESSED
0,1,0	FERRO
0,1,1	FERRO
1,0,0	RAM
1,0,1	RAM
1,1,0	ROM

B. DATA FLOW OVERVIEW

The objective of the design was to install the NM24CF04 as a functional block of memory without requiring any special actions on the part of the 8086 microprocessor. With the 8086 operating at 5 MHz and the NM24CF04 only capable of operating at 100 KHz, it was obvious that circuitry would be required to insert wait states whenever the ferroelectric memory was accessed. Additionally, circuitry was required to handle the actual transfer of data between the NM24CF04 and the 8086 microprocessor and vice-versa. Figure 2.2 shows a block diagram of the microprocessor-ferroelectric interface.

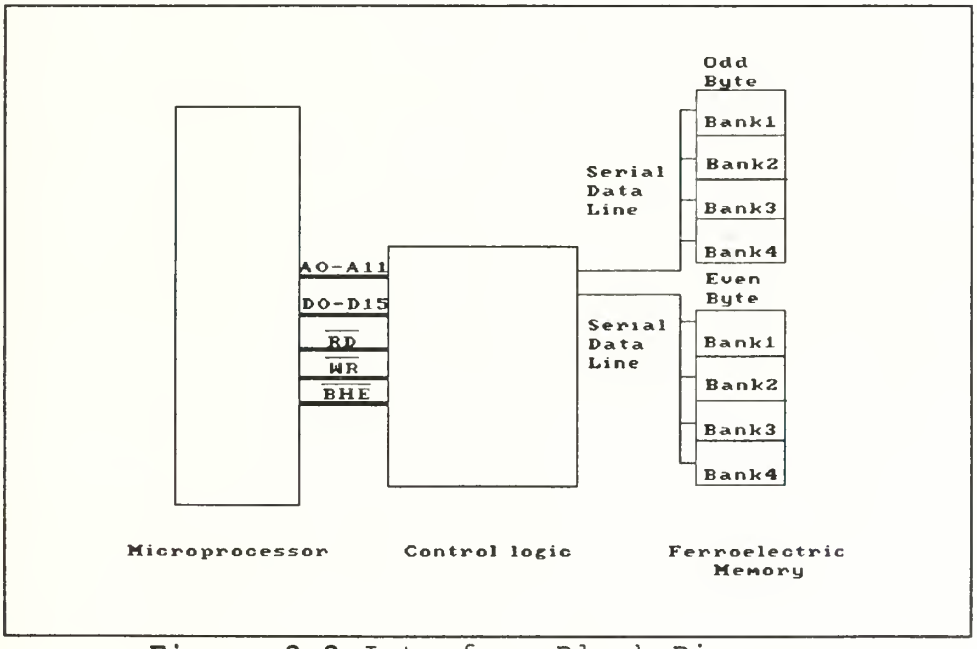


Figure 2.2 Interface Block Diagram.

The control logic monitors only five sets of signals: *RD*, *WR*, *BHE*, *A0-A15*, and *D0-D15*. When the microprocessor requests a read operation from ferroelectric memory, it places the read address on the address bus and indicates a read request by

asserting $\overline{RD}$ low. The control logic block takes over by initiating microprocessor wait states and performing the required tasks to retrieve information from the appropriate bank of ferroelectric memory. Control tasks performed include identifying and selecting the correct chip, converting the parallel address to a serial address, transmitting the address to memory serially, receiving the data from memory serially and transferring the received information into parallel form, latching the received data, and releasing control to the microprocessor. Writing to ferroelectric memory is very similar.

When the microprocessor writes data to ferroelectric memory, the write address is placed on the address bus, data on the data bus, and a write operation is indicated by asserting $\overline{WR}$ low. Again, the control block intercedes to perform data translations and input/output functions necessary to store data into the appropriate address.

C. DATA PATH DESIGN

The data path, consisting of a multiplexer and shift register, serial data (SDA) bus, ferroelectric memory, and required control signals is shown in Figure 2.3. The most important function performed by the data path is the parallel to serial conversion of information and vice versa. The data path is controlled by signals *muxa*, *muxb*, *s0*, and *s1*. Any communication with the NM24CF04 must be performed on the SDA

bus, which is bidirectional. The multiplexer in the circuit is a 4 to 1 byte multiplexer. By changing the selection lines *muxa* and *muxb*, one of four bytes can be multiplexed to the parallel load port of the shift register. The control signals *s0* and *s1* allow the shift register to be loaded in parallel from the multiplexer. After applying a clock pulse the selected byte is loaded into the shift register. A detailed discussion of the information contained in each of the four bytes is located in the control path design section.

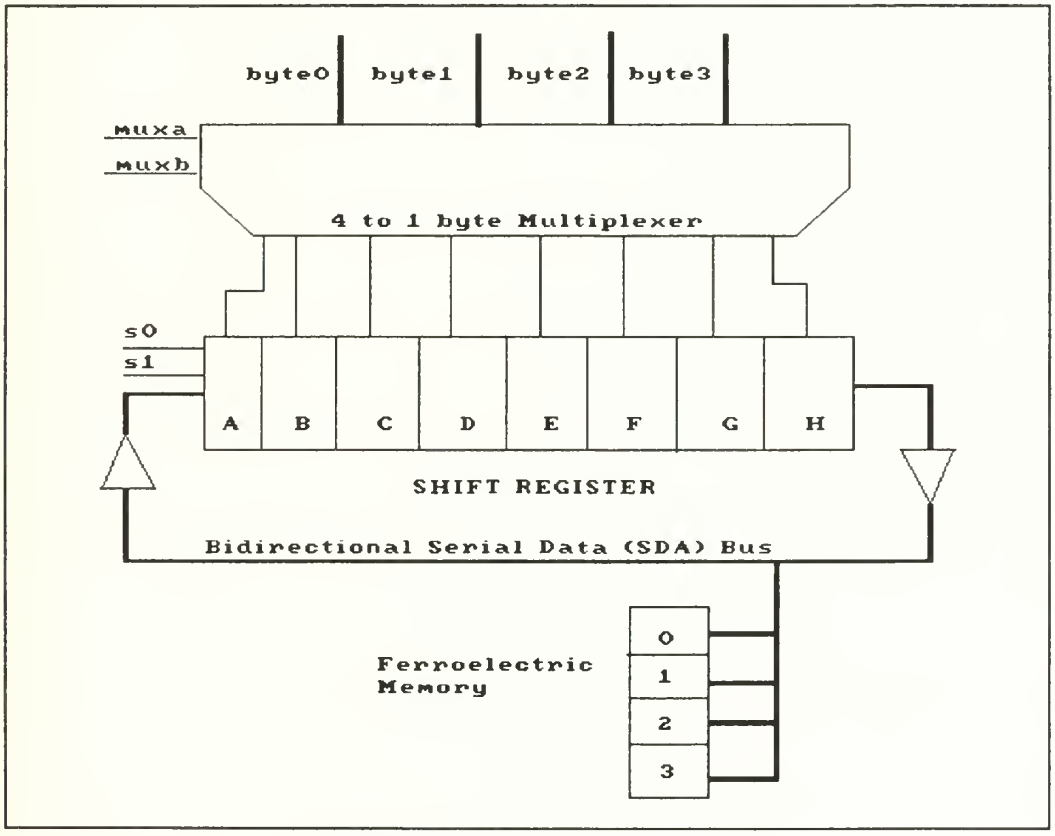


Figure 2.3 Data Path.

With information loaded, control signals *s0* and *s1* place the shift register into the right shift operation. One bit of data can be shifted to the ferroelectric memory via the SDA bus on each successive clock pulse. As soon as the parallel load has been completed, the bit in position H of the shift register is available to the ferroelectric memory when the appropriate bus driver is enabled. Therefore, the shift register only requires seven shift right operations, or clock pulses, to move all data onto the SDA bus.

When data is read from the ferroelectric memory, it becomes available on the SDA bus in serial form. Control signals *s0* and *s1* configure the shift register to shift right. The information is clocked into the shift register via another enabled bus driver. Therefore, eight clock periods are required to move a byte of data into the shift register. Once the shift register has been loaded with the received byte, the information can be latched onto the data bus.

Four ferroelectric memory banks, or separate NM24CF04 packages, can reside on the SDA bus, providing 2,048 addressable memory locations. Each bank contains 512 bytes of addressable memory. A bank is further divided into two pages of 256 bytes. A bank is designated as bank 0, 1, 2, or 3 dependent on the hardwiring of address pins upon installation. By hardwiring two address pins (A2,A1) on each package to GND or V_{cc}, a specific integrated circuit can be addressed on the common SDA bus.

D. CONTROL PATH DESIGN

The control path has the ability to control two blocks of ferroelectric memory. The blocks are referred to as the upper and lower bytes, or odd and even byte addresses, respectively. The control path provides data patterns, which contain control information, to the ferroelectric memory whenever communication is desired by the microprocessor. The data patterns are timing sensitive and are transmitted on the bidirectional SDA bus. To implement the control data patterns, with their critical timing criteria, requires two finite state machines.

1. NM24CF04 Control Bytes

To communicate with the ferroelectric memory, during a read or write operation, requires the transmission of three of four control bytes onto the SDA bus within specific timing tolerances. Whenever communication with a NM24CF04 is desired, regardless of operation, byte 0 is loaded in the shift register, via the multiplexer, and shifted onto the SDA bus. The first four bits are always a 1010 pattern to get the attention of all chips residing on the SDA bus. Again, in this design, there are four NM24CF04 integrated circuits residing on the SDA bus. The next two bits, which are from the address bus, generate a chip address corresponding to one of the hardwired addresses. This gets a specific chip's attention. The next bit is a page selection bit. Finally,

the last bit is hardwired low. Byte 1 contains the contents of the address bus lines A1 - A8 and provides the ability to address up to 512 memory locations on a specific chip. If the operation were a write, data contained in byte 2 of the multiplexer would be loaded into the shift register and transferred serially to memory. Byte 3 is used only during the write operation and is called a "dummy read". The format of the byte is identical to byte 0 except that the last bit is hardwired to +5 volts. Control byte information is summarized in Figure 2.4. Byte 2 in Figure 2.4 shows the data bus lines required to transfer data to the low block, or even byte addresses. When transferring data to the high byte, or odd byte addresses, data bus lines D8 - D16 are contained in byte 2.

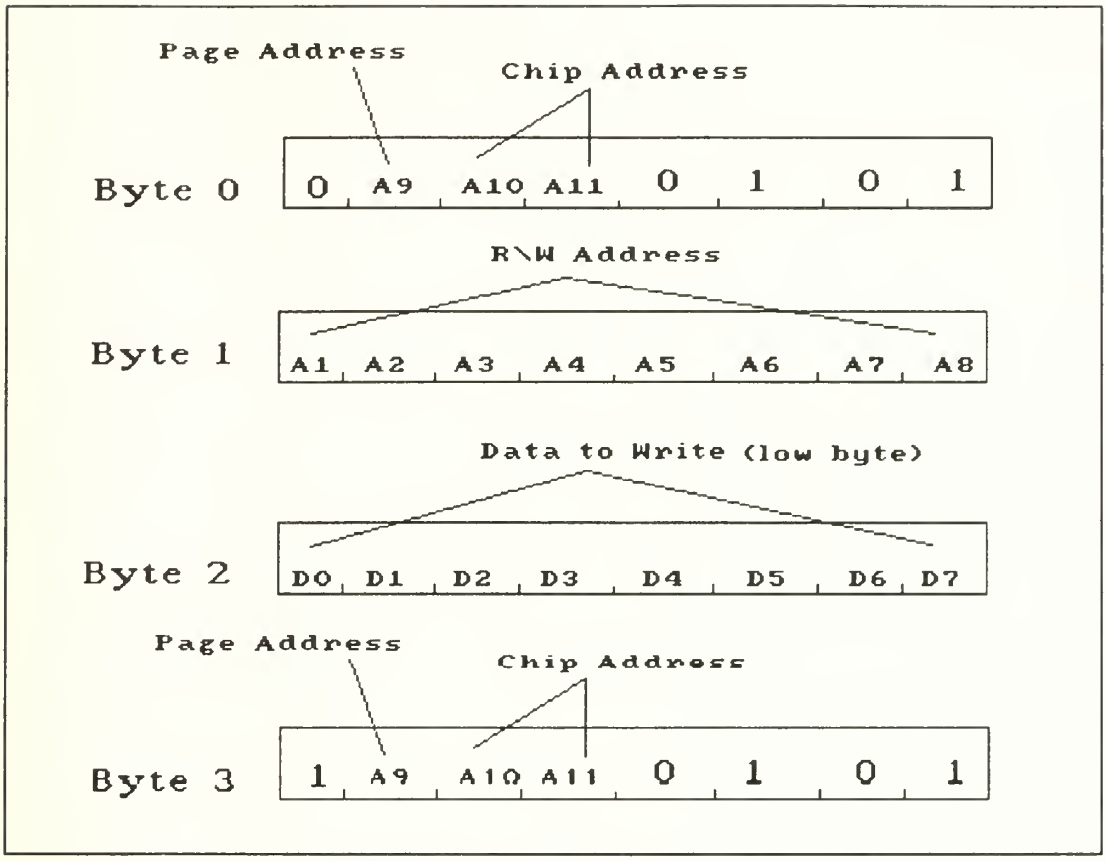


Figure 2.4 Byte Formats.

2. Main Finite State Machine

The main finite state machine is shown in Figure 2.5 and consists of 15 states. Therefore, the state machine can be produced with a four flip flop state counter, generating a count from 0 to 14. For example, in Figure 2.5, State A translates to a count of 0 and State O would translate to a count of 14 in the state counter. Notice that states B - E are performed regardless of the type of operation. The '!' symbol is used to indicate a not or low assertion of a signal.

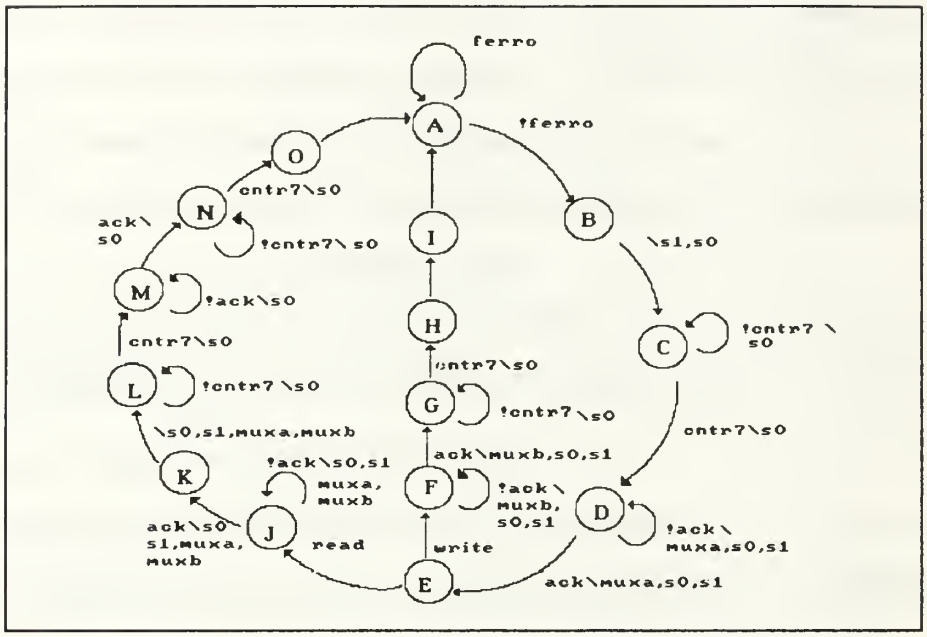


Figure 2.5 Main Finite State Machine.

STATE A: State A is the normal position of the state machine when no communication is occurring. When the microprocessor addresses the ferroelectric memory, combinational logic asserts the chip select signal *ferro* low, as indicated by *!ferro*. The state machine recognizes the chip select signal and advances the state machine to State B.

STATE B: When in this state *s0* and *s1* are asserted, which readies the shift register for a parallel load. With neither multiplexer control signals *muxa* or *muxb* asserted, byte 0 is multiplexed to the output of the byte multiplexer. Byte 0, which contains an attention pattern and chip address for the desired NM24CF04 chip residing on the SDA bus, is clocked into the shift register on the falling edge of the clock. The appropriate bus driver is enabled to allow the shift register's output onto the SDA bus.

STATE C: This state changes the mode of the shift register to shift right by only asserting *s0*. The state machine remains in this state until the shift register has been clocked right seven times. The control signal *cntr7* indicates that the shift register has been clocked seven times.

STATE D: This states looks for signal *ack*, which is transmitted on the SDA bus by an NM24CF04 when the chip address sent in byte 0 matches the hardwired chip address of the appropriate chip. While waiting for an acknowledge from the memory, *muxb*, *s0*, and *s1* are asserted to prepare the shift register for a parallel load of byte 1. Byte 1 contains the address location of where to store or retrieve data. No determination of whether a read or write operation is requested has been made yet. Again, byte 1 is loaded into the shift register on the falling edge of the clock. A bus driver is also enabled to allow the shift register's output onto the SDA bus.

STATE E: The state machine remains in this state until the shift register has been clocked right seven times as indicated by control signal *cntr7*. Once the shift register has been clocked seven times, a determination of operation is made. If the operation requested by the microprocessor is a write, the state machine will branch to State F. If a read operation is being performed, the state machine will branch to State J.

STATE F: The accessed NM24CF04 indicates that it has received the previously transmitted address by pulling the SDA bus low with signal *ack*. Control signals *s0* and *s1* are asserted, which ready the shift register for a parallel load. Multiplexer control signal *muxb* is asserted to place byte 2 on the parallel input of the shift register and the shift register is loaded on the falling edge of the clock. Byte 2 contains the data to be stored into the ferroelectric memory. Once again, bus drivers ensure the shift register has access onto the SDA bus for data transfer.

STATE G: Shift register control signal *s0* is asserted to allow the shift register to shift right. Again, the shift register is clocked right seven times as indicated by control signal *cntr7*.

STATE H: The clock is disabled on the SCL bus.

STATE I: The START_STOP flip-flop is enabled onto the SDA bus and set to generate the STOP condition for the NM24CF04. The state machine has completed a write operation and returns to State A.

STATE J: The microprocessor has requested a read operation and the address to read from was transmitted during State E. This state looks for an acknowledgement from the NM24CF04 that it has received an address. Multiplexer control signals *muxa* and *muxb* are asserted to place byte 3 onto the parallel load ports of the shift register. Byte 3 contains a "dummy write", which is identical to byte 0, except the last

bit is hardwired high. Shift register control signals *s0* and *s1* are asserted for a parallel load. Again, the shift register is clocked on the falling edge of the clock. The clock on the SCL bus is also temporarily disabled during the last half of this state.

STATE K: A START condition is transmitted onto the SDA bus, followed by activating drivers to let the shift register's least significant bit onto the SDA bus.

STATE L: Shift register control signal *s0* is asserted for a shift right operation. The shift register is clocked onto the SDA bus seven times as indicated by *cntr7*.

STATE M: The NM24CF04 acknowledges the receipt of byte 3 and places the first bit of the byte read onto the SDA bus. This bit is clocked into the shift register on the next rising edge of the clock.

STATE N: Bit one has already been shifted in, therefore, the shift register must be clocked seven more times. Again, *cntr7* indicates that seven shifts or clocks have occurred.

STATE O: The received data can be latched to the microprocessor. The SCL bus is disabled and the START_STOP is set to transmit a STOP condition to all NM24CF04's residing on the SDA bus. Control is returned to State A.

3. Counter

As previously mentioned, the second finite state machine maintains a count to control the number of shift operations performed by the shift register. The counter, when enabled, will count from zero to seven. Figure 2.6 illustrates the operation of the counter state machine. The counter is cleared by signal *counter_clr* whenever the main finite state machine is in states B, D, F, K, or M. The counter will begin counting when enabled by signal *counter_en* in states C, E, G, L, or N. While the counter is in operation, the main finite state machine is dormant and reactivated when *cntr7* is asserted by the counter.

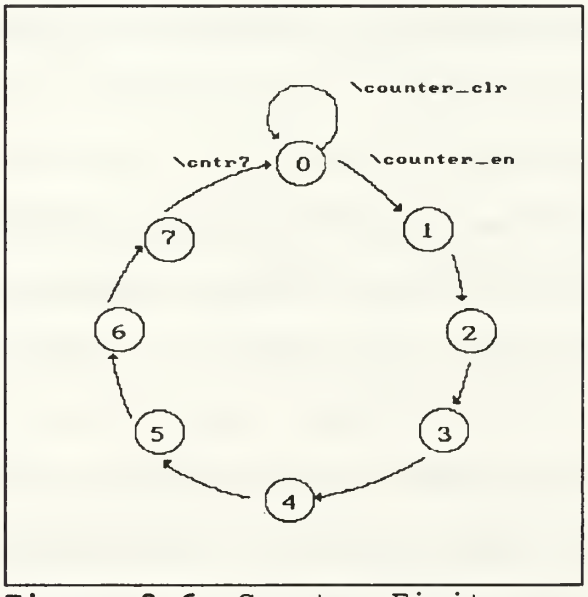


Figure 2.6 Counter Finite State Machine.

III. IMPLEMENTATION

Implementing the system described in Chapter II was quite challenging. Fifty-eight separate integrated circuits were required for the final circuit. The circuit was built on a perforated board with wire wrapped connections between chips. Most of the circuitry was required for the physical implementation of the data and control path design previously described. Ten Erasable Programmable Logic Devices (EPLD's) were utilized to reduce the number of integrated circuits required to implement much of the combinational logic for the system. Without EPLD's, the size of the system would have grown by an additional forty-five chips, causing increased power consumption. Before discussing the implementation, a more detailed description of NM24CF04's operation must be presented.

A. NM24CF04 SDA BUS OPERATION

The NM24CF04 supports a bidirectional communications protocol on the SDA bus. The occurrence of data on the SDA bus is dependent upon the phase of the clock present on the SCL bus. Data states on the SDA bus can change only on the low transition of the SCL bus. The data state must be stable on the high half of the SCL clock to allow the NM24CF04 to latch the data into its internal registers. Prior to sending

any data, a START condition must be transmitted on the SDA bus. A START condition is defined as a high to low transition of the SDA bus while SCL is high, as shown in Figure 3.1.

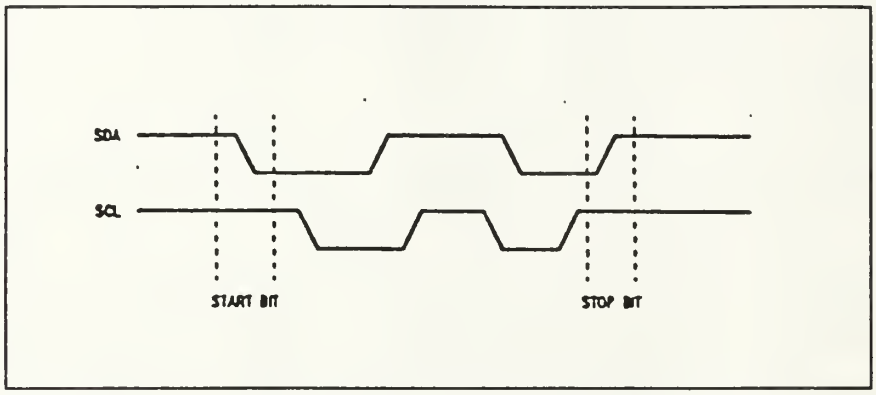


Figure 3.1 Definition of START and STOP, [Ref. 5].

Recall that both the SDA and SCL pins on the NM24CF04 are open drain devices, which require pull-up resistors and are constantly high when no external device is driving the bus. Therefore, in a working circuit, **SCL should only be clocked when the master device, in this case the microprocessor, is requesting access.** This fact is extremely important for the proper operation of the device and is not discussed in the NM24CF04 documentation. Therefore, in the implemented circuit, the clock on the SCL line was disabled whenever the main state machine was in state A. Recall the main state machine only leaves state A whenever the microprocessor requests access to the ferroelectric memory. With the clock disabled, the SCL line remains high due to the pull-up resistors. A high to low transition of the SDA bus when moving to state B satisfies the START condition. The actual

high to low transition of the SDA bus is caused by enabling a set flip-flop onto the SDA bus and then resetting it. Immediately after executing a START condition, the clock is enabled on the SCL line and the slave address is ready to be placed on the SDA bus, beginning with the first low half of the clock. The slave address consists of eight bits and the appropriate bank of memory acknowledges the slave address on the ninth clock cycle. An acknowledge is defined as the receiver, or addressed NM24CF04 in this case, pulling the SDA bus low during the ninth clock cycle. Figure 3.2 shows the timing relationship of the acknowledge response from the receiver.

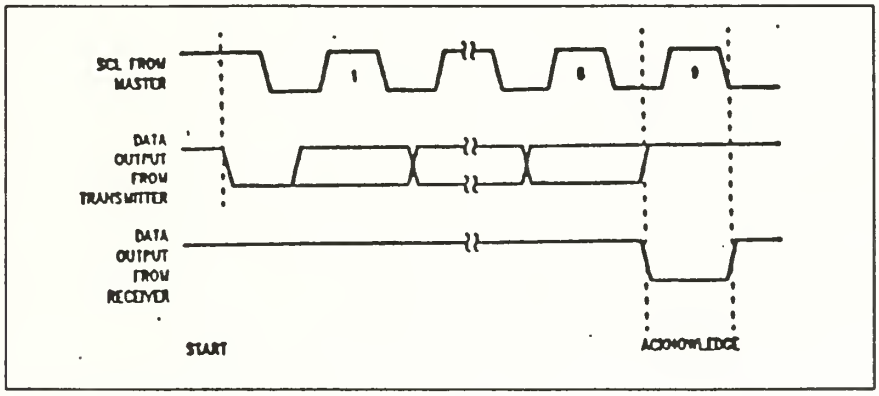


Figure 3.2 Acknowledge Response from Receiver, [Ref. 5].

After the proper device has acknowledged its address, the write or read cycle can be executed. After every byte is transmitted, the device responds with an acknowledge. Upon completion of either cycle, it is then necessary to place a STOP condition on the SDA bus. A STOP condition is defined as the low to high transition of the SDA bus while SCL is high.

This is accomplished similar to the START condition. When the cycle is completed the external clock is removed from the SCL line, which causes it to return to a high state. A reset flip-flop is then enabled onto the SDA bus and set, which effectively satisfies the STOP condition. Once the STOP condition is executed, the write or read cycle is complete and the ferroelectric memory is standing by for the next request.

B. FINITE STATE MACHINE IMPLEMENTATION

Chapter II described the design of the main finite state machine and produced a classical state diagram. Producing the state equations could have been accomplished utilizing Karnaugh maps and state reductions techniques. However, the overall design of the main finite state machine was an evolutionary process, and rather than recomputing new state equations after every change by hand the Computer Aided Design (CAD) tool PEG was used. PEG (PLA Equation Generator) is a finite state machine compiler and provides the ability to produce the appropriate state equations when a program representing the state diagram of the finite state machine is submitted for execution. PEG uses the Moore model for finite state machines, in which outputs are a function of the current state. Appendix A contains the program *FSM1.PRG,* which is the programmed representation of the main finite state machine. File *FSM1.INFO* of Appendix A is the resulting output that PEG produces after compiling *FSM1.PRG*, and contains all the state

equations in their reduced form. The main finite state machine has fifteen states. Therefore, the machine can be implemented with four flip-flops. The PEG output automatically labels these flip-flops *OutSt3*-OutSt0** and produces the state equations to control the machine. For example, looking at flip-flop *OutSt2** results in the following representation:

```
OutSt2*=
   (!RESET& cntr7& InSt0*& InSt1*&!InSt2*& InSt3*)¦
   (!RESET&!cntr7& InSt0*&!InSt1*& InSt2*& InSt3*)¦
   (!RESET&!InSt1*& InSt2*&!InSt3*)¦
   (!RESET& ack& InSt0*&!InSt1*&!InSt2*& InSt3*)¦
   (!RESET&!InSt0*& InSt1*& InSt2*&!InSt3*)¦
   (!RESET& ack&!InSt0*& InSt1*&!InSt2*& InSt3*)¦
   (!RESET&!ack&!InSt0*&!InSt1*& InSt2*& InSt3*)¦
   (!RESET&!InSt0*&!InSt1*&!InSt2*& InSt3*);
```

The symbols '¦', '!' , and '&' represent OR, NOT, and AND operations, respectively. The flip-flop represented is the second most significant bit of the machine. PEG automatically inserts the !RESET, which would be utilized if building a VLSI PLA. However, the effective reset term for the implemented finite state machine is the chip select signal *ferro*, which is low when the microprocessor requests a read or write from ferroelectric memory. Each line of the state equation, when high, can set the flip-flop on the next clock and all lines are ORed together. The resultant state equations of *FSM1.INFO* were then manually encoded into their ABEL representations and stored in program *EPLD5.ABL* of Appendix B. ABEL is an application that provides the ability to program an Altera EP310 EPLD. Flip-flops *Q4-Q1* correspond to flip-flops

OutSt3-OutSt0**, respectively, and must still be decoded by other circuitry so that specific actions might be executed depending on the current state of the machine. Continuing the example, *Q3* is the translated version of *OutSt2**, and is represented by the following equation in *EPLD5.ABL*:

```
Q3 := (!FERRO&CNTR_7&Q1&Q2&!Q3&Q4)#
      (!FERRO&!CNTR_7&Q1&!Q2&Q3&Q4)#
      (!FERRO&!Q2&Q3&!Q4)#
      (!FERRO&!Q1&Q2&Q3&!Q4)#
      (!FERRO&!Q1&!Q2&!Q3&Q4)#
      (!FERRO&ACK&((!Q1&Q2&!Q3&Q4)#(Q1&!Q2&!Q3&Q4)))#
      (!FERRO&!ACK&!Q1&!Q2&Q3&Q4);
```

The symbol '#' indicates an OR operation in ABEL. The state equations for the other flip-flops of the finite state machine were hand translated in the same manner and can be examined closer in *EPLD5.ABL*. The main finite state machine is completely encoded into one EPLD, which would normally require at least fifteen chips if standard TTL combinational logic were used. Two separate EPLD's were required to decode the state of the main finite state machine.

File *FSM1.INFO* also lists the state equations to decode states A-O. The Altera EP310 is physically limited to eight outputs. Therefore, states A-H were encoded on one EPLD and states I-O on another. The ABEL source code to implement the state decode is contained in programs *EPLD6.ABL* and *EPLD7.ABL* of Appendix B. Finally, *FSM1.INFO* contains the excitation equations to drive the multiplexer lines, *MUXA* and *MUXB*, and shift register control signals, *S0* and *S1*. These equations

are transferred to program *EPLD8.ABL* in Appendix B. This program is then downloaded to an EPLD, which provides the multiplexer and shift register control signals dependent upon the current state of the main finite state machine. The counter finite state machine was implemented in a similar manner.

The counter finite state machine's purpose is to count to seven and indicate this by asserting *CNTR_7* high. While the counter is operating the main finite state machine is in a wait state and will not re-establish control until *CNTR_7* is high. Program *FSM2.PRG* in Appendix A contains the PEG source code for the implementation of the counter finite state machine. File *FSM2.INFO* contains the result of the PEG compilation. Counting to seven only requires three flip-flops, and this is reflected in the PEG compilation by observing *OutSt2*-OutSt0**. Once again, the results of the PEG compilation were used to encode an EPLD representation of the state machine, which is contained in program *EPLD4.ABL* of Appendix B. The schematic representation of the EPLD implemented finite state machines is shown in the Ferroelectric Control Circuitry (sheet 3 of 5) of Appendix C.

C. MEMORY CHIP SELECTION

Selecting memory during a read or write cycle was approached in the classical way; combinational logic generated the appropriate chip select signal depending on the address on the address bus. All chips select signals are asserted low and generated by EPLD10, whose source code is contained in program *EPLD10.ABL* of Appendix B. The design only required 4K bytes of ferroelectric memory, 8K of ROM, and 8K of RAM. ROM consisted of two 2732A's, each of which provides 4K bytes of storage. RAM consisted of 4 AM9128's, each of which provides 2K bytes of storage. Therefore, ferroelectric memory is from address 1000H to 18FFH. RAM is mapped to address range 2000H to 28FFH. ROM is mapped to addresses 3000H and above. Each ROM chip is connected to either the upper or lower half of the data bus. When the ROM chip select signal, *ROMCS*, is asserted low by EPLD10 both banks of ROM place the addressed byte onto their halves of the data bus. If only a byte operation is required the 8086 controls the transfer by only sampling one half of the data bus. RAM is slightly more complicated.

EPLD10 provides four chip select signals for the four RAM chips: *LOWRAM1*, *LOWRAM2*, *HIRAM2*, and *HIRAM2*. The EPLD must monitor not only address lines *A12-A14*, but also *BHE* and *A0* because RAM is divided into two banks of low byte storage and two banks of high byte storage. Finally, an access of ferroelectric memory is requested when signal *FERRO* is

asserted low by EPLD10. EPLD10 also asserts *RDY1* when ferroelectric memory is accessed. Signal *RDY1* is used by the 8284 to generate wait states for the 8086 because accessing the ferroelectric memory is slow relative to the operating speed of the 8086. Table 3.1 summarizes the chip select signal asserted according to the address on the address bus.

Table 3.1 Memory Chip Select.

ADDRESS	A0	BHE	CHIP SELECT
2000-27FF	0	1	LOWRAM1
2800-2FFF	0	1	LOWRAM2
2000-27FF	1	0	HIRAM1
2800-2FFF	1	0	HIRAM2
1000-1800	X	X	FERRO
>= 3000	X	X	ROMCS

D. DATA PATH IMPLEMENTATION

The operational description of the data path was presented in Chapter II. Armed with an understanding of the functioning of the data path, a circuit was built to physically implement the data path. As described earlier, the ferroelectric memory is split into two blocks. One block is addressed by even addresses and is known as the lower byte. The other block is addressed by odd bytes and is known as the upper byte. The data path for each block is separate but identical with the exception that the upper byte reads and writes the eight most

significant bits (MSB's) of the system data bus while the lower byte reads and writes from the lower half of the system data bus. Therefore, both data paths can be implemented with duplicate circuitry. The lower data path will be discussed.

Low Ferroelectric Memory is shown in the schematics (sheet 4 of 5) contained in Appendix C. Four 74LS153's (Dual 4-to-1 Multiplexers) were chosen to provide the byte multiplexing capability required by the design. By connecting the inputs of the multiplexer chips to address lines, data lines, +5 volts, or ground the proper control byte is generated. Recall there are four unique control bytes and these appear at the output of the byte multiplexer depending on the state of signals *MUXA* and *MUXB*, which are generated by EPLD8. The control byte must then be loaded into a shift register.

The design of the data path called for a shift register that was capable of being parallel loaded or shifted right. Therefore, a 74198 Shift Register was chosen to provide this ability. The mode of the shift register is determined by the states of signals *S0* and *S1*. Both *S0* and *S1* high indicate the parallel load mode of operation, while *S0* high and *S1* low set the shift right mode. Again, *S0* and *S1* are provided by EPLD8. The shift register clock, *LO_SR_CLK*, is supplied by EPLD1. Signal *LO_SR_CLK* has two unique properties.

First, during the shift right mode, the clock is an inverted version of the 100 KHz SCL clock. This is done so that the shift register is effectively clocked on the low half

of the clock, which is required to meet the data setup and hold times of the NM24CF04 present on the SDA bus. This action is required during states C, E, G, L, M, and N.

Second, during the parallel load mode the signal provides only one low to high transition of the clock to load data into the shift register. This action is required during states B, D, F, and K of the main finite state machine.

During the shift right mode, data is shifted out *QH*. This data is be enabled onto the SDA bus. Several packages of 74LS241 Line Drivers were required to control data traffic on the SDA bus.

When the shift register is shifting data onto the SDA bus, signal *LO_SR_BUS_EN* from EPLD2 is high and drives *QH* of the shift register onto the SDA bus. After eight clocks, the driver is disabled. To receive data from memory, the shift register is configured in the shift right mode and driver signal *LO_SR_INPUT_EN* from EPLD3 is high, enabling the routing the SDA bus to the *SRSI* (shift right serial input) pin of the shift register. The shift register is clocked eight times and the data is now in parallel form. The parallel outputs of the 74LS198 *QA-QH* are now driven onto the lower half of the system data bus. The data path is therefore complete, control bytes can be transmitted to the ferroelectric memory, data can be converted to its serial representation, and finally, received data can be converted back to the parallel representation necessary for the microprocessor.

E. WRITE CYCLE IMPLEMENTATION

By decoding the address bus, signal *FERRO* (U7) is asserted low by the chip select logic of EPLD10 whenever the ferroelectric memory is accessed. The state of signal *R/W!* (U37) is the result of monitoring the $\overline{WR}$ and $\overline{RD}$ pins of the 8086; high indicates a read and low indicates a write. Both *FERRO* and *R/W!* are input to the main finite state machine, EPLD5. While *FERRO* is high the state machine remains in State A. However, when *FERRO* is low the state machine is free to start execution. Also, when *FERRO* is low, *RDY1* of EPLD10 is asserted low, which causes the 8086 to insert wait states until State I. Recall that *A0* and *BHE* are used to indicate which block of memory is being accessed. *A0* low indicates an even address or lower memory block, where *BHE* low indicates an odd address or upper memory block. Most of the control is universal regardless of which block is written to. Signals that are specifically for low block application begin with *LO*, where *HI* would be in the signal name of signals specifically for high block accesses. For example, *LO_SR_BUS_EN* (EPLD2 U17) is used only by the lower block to enable the shift register onto the lower SDA bus. There exists a signal *HI_SR_BUS_EN* (EPLD2 U17), which provides the same function for the shift register associated with the upper SDA bus. With this in mind, a write to lower memory will be discussed. Schematics to follow this discussion can be found in Appendix C.

The first task to be performed is the transmission of the START condition onto the SDA bus. A 74LS74 flip-flop (U40) is reset whenever the machine is in State A. The flip-flop is then enabled onto the SDA bus by the assertion of *LO_START_STOP_EN*, which is high for the high half of State B. Another important function that takes place when the state machine leaves State A is the enabling of the 100 KHz onto the SCL bus via the assertion of *LO_CLK_DISABLE* (EPLD4 U19). At the beginning of State B, signals *S0* and *S1* are high in preparation of a parallel load, and *MUXA* and *MUXB* are low to select the first control byte. During the low half of State B, *LO_SR_CLK* (U37) clocks the control byte into the 74198 shift register and *LO_SR_BUS_EN* enables the output of the shift register onto the lower SDA bus. During State B, the counter finite state machine located in EPLD4 is cleared by the assertion of *COUNTER_CLR* (EPLD3 U18). The main finite state machine then advances to State C, where it remains until the counter state machine has counted to seven, and indicates this by asserting *CNTR_7* (EPLD4 U19) high. While in State C, *LO_SR_CLK* continues to clock or shift the register onto the SDA bus, and *LO_SR_BUS_EN* is enabled until *CNTR_7* is asserted. Once *CNTR_7* has been detected the controller advances the machine to State D.

In State D, the controller looks for the acknowledge from memory by asserting *ACK_EN* (EPLD9 U16), which routes the SDA bus through a 7404 inverter (U38) to the input of a 74LS74

flip-flop (U40). The flip-flop is constantly clocked at 100 KHz. When the output is high, an acknowledge is interpreted by the controller, and it advances the machine to State E on the next clock. While in State D, other functions are also occurring. The counter is cleared again by asserting *COUNTER_CLR*. The shift register and multiplexer are controlled to load control byte 1. By the low half of State D, the shift register is loaded and the first bit is placed onto the SDA bus via *LO_SR_BUS_EN* again.

After advancing to State E, *LO_SR_BUS_EN* remains asserted. State E exists for seven clock cycles because the byte in the shift register must clocked onto the SDA bus. The counter is enabled by *COUNTER_EN* (EPLD3 U18), and after counting to seven asserts *CNTR_7*. After the counter has asserted *CNTR_7*, *LO_SR_BUS_EN* is disabled and a decision must be made about whether the current cycle is a read or write. The mode of the cycle is determined by the state of *R/W!*; high read or low write. For the write operation, the main finite state machine branches to State F.

In State F, an acknowledge is expected from memory for the last control byte that was transmitted. The processing of the acknowledge is just like the actions performed in State D. Again, *ACK_EN* is asserted, which routes the SDA bus through a 7404 inverter to the input of a 74LS74 flip-flop. The flip-flop is tested for acknowledge state (high), and if present the machine is advanced to State G on the next clock. State

F also clears the counter once more by asserting *COUNTER_CLR*. State F also prepares the shift register for the transfer of control byte 2, which contains the write data. Therefore, by the low half of State F, the first bit of the write data is placed onto the SDA bus via *LO_SR_BUS_EN*.

State G enables the counter and lasts for seven clocks. Again, the machine remains in this state until the remaining seven bits of write data have placed onto the SDA bus. After the assertion of *CNTR_7* by the counter *LO_SR_BUS_EN* is disabled and the machine advances to State H, where once again an acknowledge is expected. By the lower half of State H the clock is removed from the SCL bus. This is accomplished by asserting *LO_CLK_DISABLE* (EPLD4 U19) high. With the acknowledge received, the machine advances to State I.

In State I, the STOP condition must be placed onto the SDA bus. The clock has already been stopped, so the SCL is naturally high. The STOP condition is accomplished by allowing a reset 74LS74 flip-flop onto the SDA bus via *LO_START_STOP_EN*, and then setting the flip-flop with control signal *START_STOP_SET* (EPLD4 U19). The START/STOP flip-flop is common to both upper and lower memory. State I also indicates to the 8086 that the write cycle is complete by asserting *RDY1* (EPLD10 U7) high, which causes the 8086 to stop inserting wait states and continue the bus cycle. Upon completion of State I, the main finite state machine returns to State A and awaits the next access.

F. READ CYCLE IMPLEMENTATION

The read cycle begins by repeating the same actions that have already been described for States A-E of the write cycle. During State E, a check of signal *R/W!* (U37) is made. For a read operation, the signal is high. While in State E, the main finite state machine is awaiting an acknowledge from memory. Once the acknowledge has been received, via the assertion of *LO_ACK* (U40) high, the state machine branches to State J.

The disabling of the clock onto the SCL bus is the most important function performed during State J. By the lower half of State J, signal *LO_CLK_DISABLE* is asserted high, which disables a 74LS241 Line Driver (U46) and removes the 100 KHZ clock from the SCL bus. This is done to allow the SCL bus to float high in preparation for applying an additonal START condition onto the SDA bus. Additionally, *S0*, *S1*, *MUXA*, and *MUXB* are asserted high in preparation for a parallel load of control byte 3 into the 74198 Shift Register (U27). State J only lasts for one clock period and then advances to State K.

In State K, the START condition is applied to the SDA bus. Again, a reset 74LS74 flip-flop is allowed onto the SDA bus via the low assertion of *LO_START_STOP_EN*, which enables a 74LS241 Line Driver (U24). This enabling signal is only active for the high half of State K. The 100 KHz clock is enabled onto the SCL bus on the low half of State K so that data transmission can resume. By the low half of State K the

flip-flop is disabled from the SDA bus and the output of the 74198 Shift Register is enabled onto the SDA bus via the assertion of *LO_SR_BUS_EN*. It should also be pointed out that the shift register is loaded at the same time it is allowed on the bus. Recall data transitions on the low half of the clock are mandatory for proper operation of the NM24CF04. The state machine then advances to State L.

State L is active for seven clock periods once again to allow the shift register the neccessary clock cycles to shift byte 3 onto the SDA bus. Signal *S0* is asserted high to keep the shift register in the shift right mode. The main state machine remains in State L until the counter has asserted *CNTR_7* high, indicating the proper number of clock cycles have occurred. With *CNTR_7* received, the main finite state machine it free to advance to State M.

During State M, the state machine is again looking for an acknowledge from memory. The acknowledge is determined by the state of a 74LS74 flip-flop (U40) whose output generates *LO_ACK*. With *LO_ACK* high, the state machine is free to continue. Signal *S0* is asserted high during State M because read data is now expected back from the ferroelectric memory. The counter is cleared again by asserting *COUNTER_CLR*. During State M, signal *LO_SR_INPUT_EN* is asserted low enabling a bus driver (U32), and allowing the SDA bus to be applied to the *SRSI* (shift right serial input) input of the the shift register. By the lower half of State M, bit 1 from memory is

present at the input of the shift register and awaiting a clock transition on *LO_SR_CLK* to load the bit. At this time, *LO_SR_CLK* and the 100 KHz clock are synchronized and the loading of the bit and advancement of the state machine to State N occur on the next high transition of the SCL clock.

During State N, the shift register is clocked seven more times by *LO_SR_CLK* allowing the read byte to be loaded into the shift register. The counter is enabled via COUNTER_EN and counts until it reaches seven, where it then asserts *CNTR_7* again. *CNTR_7* causes the advancement of the state machine to State O and disables the 100 KHz SCL bus once again.

State O enables a bus driver (U15), which places the output of the eight stages of the shift register onto the system data bus (D0-D7). This is the read data that is now present on the system data bus, and control can now be passed back to the 8086. This is accomplished by asserting *RDY1* (EPLD10 U7) high, which is input to the 8284 (U1) and causes *READY* to go high. The 8086 ceases to generate wait states and finishes the current read cycle. A STOP condition must also be sent to the ferroelectric memory.

Recall that the SCL bus is high due to disabling of the clock at the end of State N. Therefore, a reset flip-flop (U40) is allowed onto the SDA bus under the control of *LO_START_STOP_EN*. This draws the SCL bus low. Setting the flip-flop on the next high transition of the clock satisfies the STOP condition and returns the ferroelectric memory to

standby mode, where it awaits the next access cycle. State O only lasts one clock cycle, and upon completion the state machine returns to State A.

IV. TESTING PROCEDURE

The development of the final circuit was an evolutionary process. The previous chapters of this thesis presented the final version of the design and implementation after several iterations of testing and modification. During the testing phase, many problems were discovered. Some problems were easily rectified. However, early in the testing phase, the actual functionality of the main finite state machine required further development. Unanticipated problems and errors or omissions in the manufacturer's preliminary documentation were the most prevalent reasons for modifying the design. Though at times burdensome, testing led to the eventual discovery of design and implementation errors, and finally to the goal of the research, actually reading and writing data to the NM24CF04.

A. TESTING STRATEGY

The testing strategy was to build the circuit piecemeal and verify the functionality of small sections of the implementation rather than building the entire circuit and then beginning the testing phase. This strategy split the project into smaller and more manageable parts: control circuitry, lower and upper ferroelectric memory, RAM, ROM, and

latch circuitry. The first circuitry built and tested was the control circuitry.

1. Control Circuitry

With the EPLD's programmed, the control circuitry schematic shown in Appendix C (sheet 3 of 5) was installed and verified for proper operation. Any inputs to the control circuitry such as *FERRO*, *A0*, *BHE*, *R/W!*, and *ACK* were tied either to ground or +5V. The progression through the states of the controller were verified for both the read and write operation. Surprisingly, few errors were found, and those that were found were the result of typographical errors in the source code of the EPLD program that was hand entered based on the output of the PEG compiler. To correct any problems usually only required the modification of the source program and programming a new EPLD. No logical errors were generated from the PEG compilation of the state machines. Overall, the testing of the control circuitry was a smooth process due to the efficiency and flexibility of both ABEL and PEG, which allowed the hardware design of the control circuitry to be solved more or less by software. Once the control circuitry's operation was firmly established, the installation of the ferroelectric memory was accomplished.

2. Ferroelectric Memory

Recall, that ferroelectric memory is organized into high and low blocks. The circuitry and methods to read and write data to memory are identical. Therefore, rather than building the entire circuit at once, only the lower block of memory was wire wrapped initially (sheet 4 of 5, Appendix C).

The first test to be performed was the write operation. A write only requires control byte 0-2 to perform the operation. The control bytes are provided by the byte multiplexer, which would receive address and data bus information from the 8086. The byte multiplexer had its inputs wire wrapped to ground or +5 volts to generate the required control bytes because the microprocessor was not installed at this stage of development. For example, byte 0 was wired to output the bit pattern '10101110' at the output of the byte multiplexer. The four MSB's are the required '1010' pattern that activate all chips. The next two bits contain '11', or address chip 3 on the SDA bus. The next '1' bit indicates page 1 and the last '0' bit indicates a write operation. By wiring the inputs for control bytes 1 and 2, a byte of data containing $E7_{16}$ was written to 07_{16}, or byte location 7 of the chip. However, remember that page 1 has also been selected, so byte location 263 is written to. With the inputs hardwired, a slow clock of 1 Hz was applied, and signals *FERRO*, *WR*, and *A0* were wired low. A bank of LED's

were installed to track the state of the main finite state machine and the shift operation of the shift register.

The first problem encountered was the lack of an acknowledge signal from the ferroelectric memory that was expected after transmitting the control byte 1 to the ferroelectric memory. After verifying the installed circuitry, attention was turned to the specification sheet of the NM24CF04. The concern was that since the NM24CF04 had not responded to the transmission, had the control circuitry actually started the process correctly? The START condition was studied and determined to be the culprit, or rather the timing of the START condition. In the documentation, the START Condition Setup Time was defined to be a minimum of 4.7 microseconds from the rising edge of the SCL clock. In the original implementation, the START condition was synchronized with the rising edge of the SCL clock and, therefore, did not meet the specification. After a phone conversation with the manufacturer, it was learned that the SCL clock was not required when the chip was not being accessed. Therefore, the SCL bus was floated high when the memory was not required. Then, when the START condition was applied the 4.7 microsecond requirement was satisfied. Clock disabling circuitry was added to solve the problem, and an acknowledge was finally received, which indicated that bidirectional communication had been established. The write cycle then advanced through the states of the control circuitry until the entire write cycle

was completed. Smaller problems, such as two line drivers enabled at the same time, were discovered and rectified. Figure 4.1 shows a photograph of a logic analyzer's output that resulted from monitoring the SCL and SDA bus during a successful test of the ferroelectric memory write cycle. The following is a discription of the logic analyzer output:

- Line 1 shows the monitored SCL bus. Notice periods of time when the SCL bus floats high. This is the natural state of the SCL bus when memory is not being accessed.

- Line 2 shows the activity on the monitored SDA bus. Recall a write operation requires the transmission of control bytes 0-2. Also, between each control byte is a one clock cycle delay for the acknowledge response from memory. In the test example, byte 0 contained '10101110', byte 1 contained the address of '00000111', and byte 2 contained the write data '01111110'.

- Line 3 monitors State B of the main finite state machine and was selected for clarity. Notice that State B is present for only one clock cycle. During the first half of State B, the START condition is applied to the SDA bus and during the last half of State B the clock is enabled onto the SCL bus.

- Lines 4, 5 and 6 monitor States C, G, and F of the main finite state machine. The outputs of the state machine coincide with the transmission of control bytes 0-2. Therefore, for example, Line 4 shows the duration of State C, and while State C is active, the control byte 0 bit pattern of '10101110' is displayed on the SDA bus. Likewise, lines 5 and 6 show the times control bytes 1 and 2 are on the SDA bus, respectively.

- Line 7 monitors State I of the state machine. Recall during a write operation, State I is the last state executed. Notice the SCL clock is disabled, signalling the end of the write cycle.

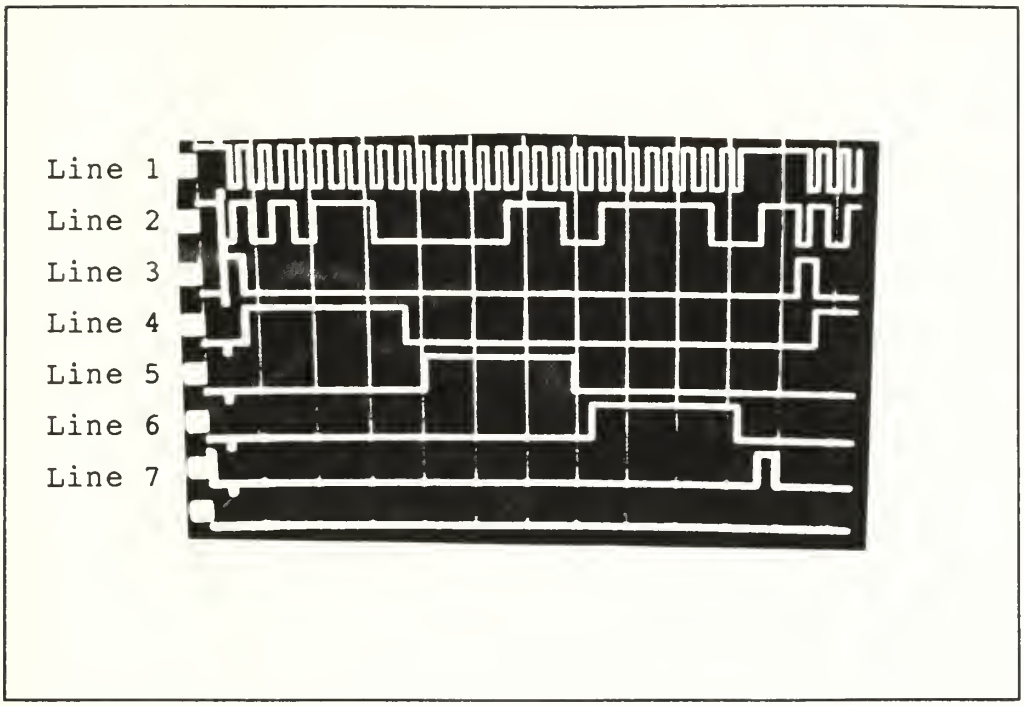


Figure 4.1 Ferroelectric Memory Write Cycle.

With confidence in the write operation attention was turned to the read cycle. As a test an attempt was made to read the data out of the same memory location that had just been written to. The read cycle duplicates States A-E of the write cycle, therefore, half of the work was already done and confirmed. States J-O required validation and once again timing errors, such as drivers being on for one half a clock period too long, seemed to predominate the testing process. Finally, all corrections were made via new EPLD programming or additional circuitry, and a valid read cycle was performed. Figure 4.2 shows the logic analyzer's output of the successful read operation.

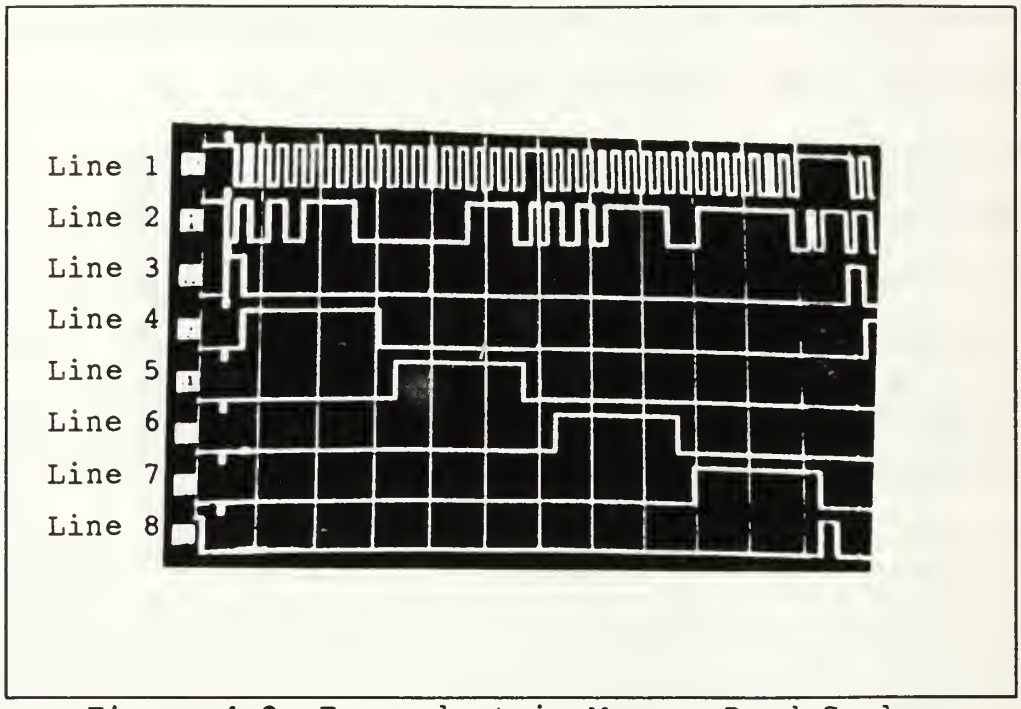


Figure 4.2 Ferroelectric Memory Read Cycle.

The first five lines of the photograph are identical to the write cycle presented in Figure 4.1. However, the read cycle lasts seventeen clock cycles longer, therefore, the sampling period of the logic analyzer had to be changed to allow the entire read cycle to be displayed on the screen. Once again, the bottom three lines were added to show the timing of the SDA bus relative to the state of the controller. Notice that at the end of State G, the clock on the SCL bus is disabled. This is required because of the second START condition that must be placed on the SDA bus during a read operation. Line 6 indicates State L, and the activity on the SDA bus at this time is due to the transmission of control byte 1. Line 7 corresponds to State N, which is when the

received data is clocked into the shift register. The SDA bus during this time period shows that a '01111110'is present, which is the data that was written to the chip during the write test. Line 8 of Figure 4.2 shows the occurrance of State O, which idicates the end of the read cycle. Notice that during State O, the SCL bus is floated high, and the low to high transition of the STOP condition is placed on the SDA bus.

The nonvolatility of the data was the final test. Power was removed from the circuit for a day. The circuit was activated and the read test was performed again. The data pattern stored the previous day was retrieved, which proved the manufacturer's claim of nonvolatility. The specification sheet claims 10 years of data retention. Obviously, this specification can not be tested during the research time of this thesis. Testing was also performed to validate some of the specifications contained in Reference 5. First the timing specifications of the NM24CF04 were tested.

Read and write cycles were performed at various clock frequencies. The NM24CF04 had no problems operating at clock frequencies at or below the 100 KHZ specification. However, at frequencies above 100 KHz, the chip did not function properly. It usually missed the attention signal from control byte 0, or placed useless information on the SDA bus when operating at the higher frequencies.

The manufacturer also claims a write cycle of less than 200 nanoseconds. The results of the write test show that 29 clock cycles are required for a write cycle. At 100KHz the period of the clock is 10 microseconds. Therefore, the overall time requirement for the write operation is 290 microseconds. The discrepancy between the testing results and the NM24CF04 specifications could be a matter of write cycle definition.

This study considers the write cycle time as the time that passes from the point when the 8086 places the write address on the address bus, which starts the main finite state machine, to the time when the finite state machine returns to State A. The data sheet claim of less than 200 nanoseconds for a write cycle could be the time required to latch data into an individual memory cell.

The results of the read test show that 38 clock cycles are required for the read operation. Therefore, a read cycle requires 380 microseconds.

V. CONCLUSIONS

This study has successfully demonstrated the feasibility of interfacing the NM24CF04 as a block of 8086 microprocessor main memory. The use of ferroelectric technology would provide a radiation hardened and nonvolatile, yet modifiable, area of memory where mission parameters or new programs could be stored. The nonvolatility of the NM24CF04 has been proven. The use of this new memory device in any system would combine the strengths of current technologies; the flexibility of RAM and the nonvolatility of ROM. However, the data and address communications, timing, and control for the ferroelectric memory is significantly different than for standard ROM and RAM.

This research has successfully developed a hardware solution to overcome the communications, timing, and control difficulties of the bidirectional protocol necessary to access the NM24CF04. The bidirectional protocol is somewhat cumbersome and places additional delays on an already slow device, relative to the 5 MHz operating speed of the 8086.

For example, the write cycle takes 290 microseconds to complete. With the 8086 microprocessor operating at 5MHz, or a clock period of 200 nanoseconds, this would require the 8086 microprocessor to insert 1450 wait states before the data was latched into the NM24CF04. The read cycle requires 380

microseconds, and the microprocessor would insert 1900 wait states before data would be returned from the ferroelectric memory. With faster microprocessors, the number of wait states would increase significantly. As the ferroelectric technology evolves, what is really needed is a ferroelectric integrated circuit that is accessed by means compatible with current RAM accessing techniques. The availability of a parallel accessed ferroelectric memory would decrease the read and write cycle times.

Counteracting the speed concerns of this integrated circuit is the improvement in the number of write cycles when compared to the EEPROM version of the NM24CF04. With 10^{11} write cycles, approximately 22 years of constant writes could be performed on the integrated circuit, as derived in the following equation:

$$1*10^{11}*290\frac{microsecs}{1write}*1\frac{\text{sec}}{1*10^{6}microsecs}*1\frac{\text{min}}{60secs}*1\frac{hr}{60mins}*$$

$$1\frac{day}{24hrs}*1\frac{yr}{365days}=22.07yrs$$

Using the same equation, except with the number of write cycles reduced to 10^5, the EEPROM version of the chip would reach the write cycle limit after only 29 seconds of constantly writing to the chip. Clearly, the ferroelectric version of the integrated circuit shows a most noteworthy

performance improvement. Of course, the microprocessor is not going to constantly access the ferroelectric memory, but the ability to perform over a long time period adds futher credibilty to the assertion that the NM24CF04 become part of main memory that has historically been reserved for RAM and ROM.

APPENDIX A
PEG PROGRAMS

A. PROGRAM 'FSM1.PRG'

```
INPUTS:  RESET ferro cntr7 ack wr;
OUTPUTS: s1 s0 muxa muxb a b c d e f g h i j k l m n o;

start:   ASSERT a;
         IF NOT ferro THEN stateb ELSE LOOP;

stateb:  ASSERT b s1 s0;
         GOTO statec;

statec:  ASSERT c s0;
         IF cntr7 THEN  stated ELSE LOOP;

stated:  ASSERT d muxa s0 s1;
         IF ack THEN  statee ELSE LOOP;

statee:  ASSERT e s0;
         CASE (cntr7 wr)
         1 0 => statef;
         1 1 => statej;
         ENDCASE => statee;

statef:  ASSERT f s0 s1 muxb;
         IF ack THEN  stateg ELSE LOOP;

stateg:  ASSERT g s0;
         IF cntr7 THEN stateh ELSE LOOP;

stateh:  ASSERT h ;
         GOTO statei;

statei:  ASSERT i;
         GOTO start;

statej:  ASSERT j s0 s1 muxa muxb;
         IF ack THEN statek ELSE LOOP;

statek:  ASSERT k s0 s1 muxa muxb;
         GOTO statel;

statel:  ASSERT l s0;
         IF cntr7 THEN statem ELSE LOOP;
```

```
statem:  ASSERT m s0;
         IF ack THEN staten ELSE LOOP;

staten:  ASSERT n s0;
         IF cntr7 THEN stateo ELSE LOOP;

stateo:  ASSERT o;
         GOTO start;
```

B. FILE 'FSM1.INFO'

```
INORDER=
   RESET
   ferro
   cntr7
   ack
   wr
   InSt0*
   InSt1*
   InSt2*
   InSt3*;
OUTORDER=
   OutSt3*
   OutSt2*
   OutSt1*
   OutSt0*
   s1
   s0
   muxa
   muxb
   a
   b
   c
   d
   e
   f
   g
   h
   i
   j
   k
   l
   m
   n
   o;
OutSt3*=
     (!RESET&!cntr7& InSt0*& InSt1*&!InSt2*& InSt3*)¦
     (!RESET& ack& InSt0*& InSt1*&!InSt2*&!InSt3*)¦
     (!RESET&!cntr7& InSt0*&!InSt1*& InSt2*& InSt3*)¦
     (!RESET& InSt0*&!InSt1*& InSt2*&!InSt3*)¦
     (!RESET&!ack& InSt0*&!InSt1*&!InSt2*& InSt3*)¦
     (!RESET& cntr7&!InSt0*& InSt2*&!InSt3*)¦
     (!RESET&!ack&!InSt0*& InSt1*&!InSt2*& InSt3*)¦
     (!RESET& cntr7&!InSt0*& InSt1*&!InSt2*&!InSt3*)¦
     (!RESET&!ack&!InSt0*&!InSt1*& InSt2*& InSt3*)¦
     (!RESET&!ferro&!InSt0*&!InSt1*&!InSt2*&!InSt3*);
```

```
OutSt2*=
     (!RESET& cntr7& InSt0*& InSt1*&!InSt2*& InSt3*)¦
     (!RESET&!cntr7& InSt0*&!InSt1*& InSt2*& InSt3*)¦
     (!RESET&!InSt1*& InSt2*&!InSt3*)¦
     (!RESET& ack& InSt0*&!InSt1*&!InSt2*& InSt3*)¦
     (!RESET&!InSt0*& InSt1*& InSt2*&!InSt3*)¦
     (!RESET& ack&!InSt0*& InSt1*&!InSt2*& InSt3*)¦
     (!RESET&!ack&!InSt0*&!InSt1*& InSt2*& InSt3*)¦
     (!RESET&!InSt0*&!InSt1*&!InSt2*& InSt3*);
OutSt1*=
     (!RESET& InSt0*& InSt1*&!InSt2*)¦
     (!RESET& cntr7& InSt0*&!InSt1*& InSt2*& InSt3*)¦
     (!RESET&!InSt0*& InSt1*& InSt2*&!InSt3*)¦
     (!RESET&!InSt0*& InSt1*&!InSt2*& InSt3*)¦
     (!RESET& cntr7&!wr&!InSt0*& InSt1*&!InSt2*&!InSt3*)¦
     (!RESET&!cntr7&!InSt0*& InSt1*&!InSt2*&!InSt3*)¦
     (!RESET& ack&!InSt0*&!InSt1*& InSt2*& InSt3*);
OutSt0*=
     (!RESET& InSt0*& InSt1*&!InSt2*)¦
     (!RESET& InSt0*&!InSt1*& InSt2*)¦
     (!RESET& InSt0*&!InSt1*&!InSt2*& InSt3*)¦
     (!RESET&!InSt0*& InSt1*& InSt2*& InSt3*)¦
     (!RESET& cntr7& wr&!InSt0*& InSt1*&!InSt2*&!InSt3*);
s1=
     ( InSt0*&!InSt1*& InSt2*&!InSt3*)¦
     (!InSt1*&!InSt2*& InSt3*)¦
     (!InSt0*& InSt1*&!InSt2*& InSt3*)¦
     (!InSt0*&!InSt1*& InSt2*& InSt3*);
s0=
     ( InSt0*& InSt1*&!InSt2*)¦
     (!InSt1*& InSt2*)¦
     (!InSt1*&!InSt2*& InSt3*)¦
     (!InSt0*& InSt1*&!InSt3*)¦
     (!InSt0*& InSt1*&!InSt2*& InSt3*);
muxa=
     ( InSt0*&!InSt1*& InSt2*&!InSt3*)¦
     ( InSt0*&!InSt1*&!InSt2*& InSt3*)¦
     (!InSt0*&!InSt1*& InSt2*& InSt3*);
muxb=
     ( InSt0*&!InSt1*& InSt2*&!InSt3*)¦
     ( InSt0*&!InSt1*&!InSt2*& InSt3*)¦
     (!InSt0*& InSt1*&!InSt2*& InSt3*);
a=
     (!InSt0*&!InSt1*&!InSt2*&!InSt3*);
b=
     (!InSt0*&!InSt1*&!InSt2*& InSt3*);
c=
     (!InSt0*&!InSt1*& InSt2*&!InSt3*);

d=
     (!InSt0*&!InSt1*& InSt2*& InSt3*);
```

```
e=
     (!InSt0*& InSt1*&!InSt2*&!InSt3*);
f=
     (!InSt0*& InSt1*&!InSt2*& InSt3*);
g=
     (!InSt0*& InSt1*& InSt2*&!InSt3*);
h=
     (!InSt0*& InSt1*& InSt2*& InSt3*);
i=
     ( InSt0*&!InSt1*&!InSt2*&!InSt3*);
j=
     ( InSt0*&!InSt1*&!InSt2*& InSt3*);
k=
     ( InSt0*&!InSt1*& InSt2*&!InSt3*);
l=
     ( InSt0*&!InSt1*& InSt2*& InSt3*);
m=
     ( InSt0*& InSt1*&!InSt2*&!InSt3*);
n=
     ( InSt0*& InSt1*&!InSt2*& InSt3*);
o=
     ( InSt0*& InSt1*& InSt2*&!InSt3*);
```

C. PROGRAM 'FSM2.PRG'

```
INPUTS: RESET ;
OUTPUTS: cntr7 ;

start:
     :
     :
     :
     :
     :
     :
     : ASSERT cntr7;
       GOTO start;
```

D. FILE 'FSM2.INFO'

```
INORDER=
   RESET
   InSt0*
   InSt1*
   InSt2*;
OUTORDER=
   OutSt2*
   OutSt1*
   OutSt0*
   cntr7;
OutSt2*=
     (!RESET&!InSt2*);
OutSt1*=
     (!RESET& InSt1*&!InSt2*)¦
     (!RESET&!InSt1*& InSt2*);
OutSt0*=
     (!RESET& InSt0*&!InSt2*)¦
     (!RESET& InSt0*&!InSt1*& InSt2*)¦
     (!RESET&!InSt0*& InSt1*& InSt2*);
cntr7=
     ( InSt0*& InSt1*& InSt2*);
```

APPENDIX B
ERASEABLE PROGRAMMABLE LOGIC DEVICE PROGRAMS

A. PROGRAM 'EPLD1.ABL'

```
MODULE EPLD1
TITLE 'SHIFT REGISTER CLOCK ENABLE 08/09/91'
"AUTHOR -- T. C. GONTER

     U1 DEVICE 'E0310';

"Input pins

     CLK                 PIN 1;            "100 KHZ CLOCK
     B,C,D,E             PIN 2,3,4,5;      "STATES
     F,G,K,L,M           PIN 6,7,8,9,11;   "STATES
     A0,BHE              PIN 12,13;        "8086 CONTROL LINES
     N                   PIN 14;           "FSM #1 STATES
"Output pins

     LO_SR_LOAD_CLK      PIN 19;          "LOW SHIFT REGISTER
                                          "CLOCK
     LO_SR_LOAD_CLK      IsType 'com,feed_or,pos';

     LO_SR_SHIFT_CLK     PIN 18;          "LOW SHIFT REGISTER
                                          "CLOCK
     LO_SR_SHIFT_CLK     IsType 'com,feed_or,pos';

     HI_SR_LOAD_CLK      PIN 17;         "HIGH SHIFT REGISTER
                                         "CLOCK
     HI_SR_LOAD_CLK      IsType 'com,feed_or,pos';

     HI_SR_SHIFT_CLK     PIN 16;         "HIGH SHIFT REGISTER
                                         "CLOCK
     HI_SR_SHIFT_CLK     IsType 'com,feed_or,pos';

"Equivalences

     X =.X.;
     STATES = [B,C,D,F,E,G,K,L,M,N,X,X];
```

```
EQUATIONS
                                                "CLK SR ON LOW TRANS
                                                "OF CLK WHEN DOING
                                                "PARALLEL LOAD.
                                                "LOAD ON B,D,F,K

    LO_SR_LOAD_CLK = !A0&!CLK&((STATES==^H200)#
                     (STATES==^H800)#
                     (STATES==^H100)#
                     (STATES==^H020));
                                                "CLK SR ON LOW TRANS
                                                "OF CLK WHEN SHIFTING.
                                                "SHIFT ON C,E,G,L

    LO_SR_SHIFT_CLK = !A0&((!CLK&((STATES==^H080)#
                       (STATES==^H040)#
                       (STATES==^H010)#
                       (STATES==^H400)))#
                       (CLK&((STATES==^H004)#
                             (STATES ==^H008))));

                                                "CLK SR ON LOW TRANS
                                                "OF CLK WHEN DOING
                                                "PARALLEL LOAD.
                                                "LOAD ON B,D,F,K

    HI_SR_LOAD_CLK = !BHE&!CLK&((STATES==^H200)#
                     (STATES==^H800)#
                     (STATES==^H100)#
                     (STATES==^H020));
                                                "CLK SR ON LOW TRANS
                                                "OF CLK WHEN SHIFTING.
                                                "SHIFT ON C,E,G,L,M

    HI_SR_SHIFT_CLK = !BHE&((!CLK&((STATES==^H080)#
                       (STATES==^H040)#
                       (STATES==^H010)#
                       (STATES==^H400)))#
                       (CLK&((STATES==^H004)#
                             (STATES ==^H008))));

TEST_VECTORS ([CLK,STATES,A0,BHE] ->

[LO_SR_LOAD_CLK,LO_SR_SHIFT_CLK,HI_SR_LOAD_CLK,HI_SR_SHIFT_C
LK])
                    [0, ^H200,0,1]   -> [1,0,0,0];
                    [0, ^H800,0,1]   -> [1,0,0,0];
                    [0, ^H100,0,1]   -> [1,0,0,0];
                    [0, ^H020,0,1]   -> [1,0,0,0];
                    [1, ^H200,0,1]   -> [0,0,0,0];
                    [1, ^H800,0,1]   -> [0,0,0,0];
```

```
                    [1, ^H100,0,1]   -> [0,0,0,0];
                    [1, ^H020,0,1]   -> [0,0,0,0];
                    [0, ^H008,0,1]   -> [0,0,0,0];
                    [0, ^H080,0,1]   -> [0,1,0,0];
                    [0, ^H040,0,1]   -> [0,1,0,0];
                    [0, ^H010,0,1]   -> [0,1,0,0];
                    [0, ^H400,0,1]   -> [0,1,0,0];
                    [0, ^H004,0,1]   -> [0,0,0,0];
                    [1, ^H008,0,1]   -> [0,1,0,0];
                    [1, ^H080,0,1]   -> [0,0,0,0];
                    [1, ^H040,0,1]   -> [0,0,0,0];
                    [1, ^H010,0,1]   -> [0,0,0,0];
                    [1, ^H400,0,1]   -> [0,0,0,0];
                    [1, ^H004,0,1]   -> [0,1,0,0];
                    [0, ^H200,1,0]   -> [0,0,1,0];
                    [0, ^H800,1,0]   -> [0,0,1,0];
                    [0, ^H100,1,0]   -> [0,0,1,0];
                    [0, ^H020,1,0]   -> [0,0,1,0];
                    [1, ^H200,1,0]   -> [0,0,0,0];
                    [1, ^H800,1,0]   -> [0,0,0,0];
                    [1, ^H100,1,0]   -> [0,0,0,0];
                    [1, ^H020,1,0]   -> [0,0,0,0];
                    [0, ^H008,1,0]   -> [0,0,0,0];
                    [0, ^H080,1,0]   -> [0,0,0,1];
                    [0, ^H040,1,0]   -> [0,0,0,1];
                    [0, ^H010,1,0]   -> [0,0,0,1];
                    [0, ^H400,1,0]   -> [0,0,0,1];
                    [0, ^H004,1,0]   -> [0,0,0,0];
                    [1, ^H008,1,0]   -> [0,0,0,1];
                    [1, ^H080,1,0]   -> [0,0,0,0];
                    [1, ^H040,1,0]   -> [0,0,0,0];
                    [1, ^H010,1,0]   -> [0,0,0,0];
                    [1, ^H400,1,0]   -> [0,0,0,0];
                    [1, ^H004,1,0]   -> [0,0,0,1];

END EPLD1
```

B. PROGRAM 'EPLD2.ABL'

```
MODULE EPLD2
FLAG '-r2'
TITLE 'SHIFT REGISTER OUTPUT BUS ENABLE 07/31/91'

"AUTHOR: T. C. GONTER

     U1 DEVICE 'E0310';

"Input pins

     CLK                 PIN 1;          "100 KHZ CLOCK
     C,E,G,L             PIN 2,3,4,5;    "FINITE STATE INPUTS
     B,D,F,K             PIN 6,7,8,9;    "FINITE STATE INPUTS
     A0,BHE              PIN 11,12;      "8086 CONTROL LINES
     CNTR_7              PIN 13;         "SHIFT COUNT

"Output pins

     LO_SR_BUS_EN          PIN 19;        "LOW SHIFT REG BUS
                                           "ENABLE
     LO_SR_BUS_EN           IsType 'com,feed_or,pos';

     HI_SR_BUS_EN          PIN 18;        "HIGH SHIFT REG BUS
                                           "ENABLE
     HI_SR_BUS_EN           IsType 'com,feed_or,pos';

     RTERM1                 PIN 17;        "REDUCTION TERM
     RTERM1                 IsType 'com,feed_or,pos';

     RTERM2                 PIN 16;        "REDUCTION TERM
     RTERM2                 IsType 'com,feed_or,pos';

     RTERM3                 PIN 15;        "REDUCTION TERM
     RTERM3                 IsType 'com,feed_or,pos';

"Equivalences
     STATES = [C,E,G,L];

EQUATIONS
                                           "ENABLE LO SR
                                           "OUTPUT ON SDA BUS

     LO_SR_BUS_EN = !A0&(RTERM3#RTERM2#RTERM1);
```

```
                                                    "ENABLE HI SR
                                                    "OUTPUT ON SDA BUS

HI_SR_BUS_EN = !BHE&(RTERM3#RTERM2#RTERM1);

RTERM1 = (!CLK&(B # D # F # K));

RTERM2 = ((C$(C&CNTR_7&!CLK))#
          (E$(E&CNTR_7&!CLK))#
          (G$(G&CNTR_7&!CLK)));

RTERM3 = (L$(L&CNTR_7&!CLK));

TEST_VECTORS    ([A0,BHE,CNTR_7,C,E,G,L,!CLK,B,D,F,K] ->
                                  [LO_SR_BUS_EN,HI_SR_BUS_EN])
            [0,0,0,0,0,0,0,0,0,0,0,0]    ->   [ 0,0 ];
            [0,1,0,1,0,0,0,1,0,0,0,0]    ->   [ 1,0 ];
            [0,1,0,1,0,0,0,0,0,0,0,0]    ->   [ 1,0 ];
            [0,1,1,1,0,0,0,0,0,0,0,0]    ->   [ 1,0 ];
            [0,1,1,1,0,0,0,1,0,0,0,0]    ->   [ 0,0 ];
            [0,1,0,0,1,0,0,0,0,0,0,0]    ->   [ 1,0 ];
            [0,1,0,0,1,0,0,1,0,0,0,0]    ->   [ 1,0 ];
            [0,1,0,0,0,1,0,0,0,0,0,0]    ->   [ 1,0 ];
            [0,1,0,0,0,1,0,1,0,0,0,0]    ->   [ 1,0 ];
            [0,1,0,0,0,0,1,0,0,0,0,0]    ->   [ 1,0 ];
            [0,1,0,0,0,0,1,1,0,0,0,0]    ->   [ 1,0 ];
            [0,1,0,0,0,0,0,1,0,0,0,0]    ->   [ 0,0 ];
            [0,1,0,0,0,0,0,1,1,0,0,0]    ->   [ 1,0 ];
            [0,1,0,0,0,0,0,1,0,1,0,0]    ->   [ 1,0 ];
            [0,1,0,0,0,0,0,1,0,0,1,0]    ->   [ 1,0 ];
            [0,1,0,0,0,0,0,1,0,0,0,1]    ->   [ 1,0 ];
            [0,1,0,0,0,0,0,0,1,0,0,0]    ->   [ 0,0 ];
            [0,1,0,0,0,0,0,0,0,1,0,0]    ->   [ 0,0 ];
            [0,1,0,0,0,0,0,0,0,0,1,0]    ->   [ 0,0 ];
            [0,1,0,0,0,0,0,0,0,0,0,1]    ->   [ 0,0 ];
            [0,0,0,0,0,0,0,0,0,0,0,0]    ->   [ 0,0 ];
            [1,0,0,1,0,0,0,1,0,0,0,0]    ->   [ 0,1 ];
            [1,0,0,1,0,0,0,0,0,0,0,0]    ->   [ 0,1 ];
            [1,0,1,1,0,0,0,0,0,0,0,0]    ->   [ 0,1 ];
            [1,0,0,0,1,0,0,0,0,0,0,0]    ->   [ 0,1 ];
            [1,0,0,0,1,0,0,1,0,0,0,0]    ->   [ 0,1 ];
            [1,0,0,0,0,1,0,0,0,0,0,0]    ->   [ 0,1 ];
            [1,0,0,0,0,1,0,1,0,0,0,0]    ->   [ 0,1 ];
            [1,0,0,0,0,0,1,0,0,0,0,0]    ->   [ 0,1 ];
            [1,0,0,0,0,0,1,1,0,0,0,0]    ->   [ 0,1 ];
            [1,0,0,0,0,0,0,1,0,0,0,0]    ->   [ 0,0 ];
            [1,0,0,0,0,0,0,1,1,0,0,0]    ->   [ 0,1 ];
            [1,0,0,0,0,0,0,1,0,1,0,0]    ->   [ 0,1 ];
            [1,0,0,0,0,0,0,1,0,0,1,0]    ->   [ 0,1 ];
            [1,0,0,0,0,0,0,1,0,0,0,1]    ->   [ 0,1 ];
            [1,0,0,0,0,0,0,0,1,0,0,0]    ->   [ 0,0 ];
```

```
                [1,0,0,0,0,0,0,0,1,0,0]    ->  [ 0,0 ];
                [1,0,0,0,0,0,0,0,0,1,0]    ->  [ 0,0 ];
                [1,0,0,0,0,0,0,0,0,0,1]    ->  [ 0,0 ];

END EPLD2
```

C. PROGRAM 'EPLD3.ABL'

```
MODULE EPLD3
TITLE 'FSM #2 -- COUNTER CONTROL 06/10/91'

     U1 DEVICE 'E0310';

"Input pins

     B,D,F,K              PIN 2,3,4,5;          "FINITE STATE
                                                "INPUTS

     L,C,E,G,M,N          PIN 6,7,8,9,11,12;    "FINITE STATE
                                                "INPUTS

"Output pins

     COUNTER_CLR          PIN 19;               "COUNTER CLEAR

     COUNTER_EN           PIN 18;               "COUNTER ENABLE

EQUATIONS

     COUNTER_CLR = B#D#F#K#M;

     COUNTER_EN = C#E#G#L#N;

    TEST_VECTORS   ([B,D,F,K,M,C,E,G,L,N] ->
                                  [COUNTER_CLR,COUNTER_EN])
                    [0,0,0,0,0,0,0,0,0,0] ->     [0,0];
                    [1,0,0,0,0,0,0,0,0,0] ->     [1,0];
                    [0,1,0,0,0,0,0,0,0,0] ->     [1,0];
                    [0,0,1,0,0,0,0,0,0,0] ->     [1,0];
                    [0,0,0,1,0,0,0,0,0,0] ->     [1,0];
                    [0,0,0,0,1,0,0,0,0,0] ->     [1,0];
                    [0,0,0,0,0,1,0,0,0,0] ->     [0,1];
                    [0,0,0,0,0,0,1,0,0,0] ->     [0,1];
                    [0,0,0,0,0,0,0,1,0,0] ->     [0,1];
                    [0,0,0,0,0,0,0,0,1,0] ->     [0,1];
                    [0,0,0,0,0,0,0,0,0,1] ->     [0,1];

 END EPLD3
```

D. PROGRAM 'EPLD4.ABL'

```
MODULE EPLD4
FLAG '-r2'
TITLE 'FSM #2 -- BIT SHIFT COUNTER 07/31/91'

     U1 DEVICE 'E0310';

"Input pins

     CLK                  PIN 1;              "100 KHZ CLOCK
     COUNTER_CLR          PIN 2;              "CLEAR COUNTER
     COUNTER_EN           PIN 3;              "ENABLE COUNTER
     N,A,H,J,I,O          PIN 4,5,6,7,8,9;    "FSM #1 STATES
     Q1,Q2,Q3             PIN 14,15,16;       "COUNTER
     Q1,Q2,Q3             IsType 'feed_reg,reg_d,pos';
     BHE,A0              PIN 12,11;          "8086 CONTROL LINES

"Output pins

     CNTR_7               PIN 19;             "COUNTER = 7
     CNTR_7               IsType 'com,feed_or,pos';

     STOP_START_SET       PIN 18;            "SET STOP_START FF
     STOP_START_SET       IsType 'com,feed_or,pos';

     LO_CLK_DISABLE       PIN 13;             "LO FERRO SCL
                                              "ENABLE
     LO_CLK_DISABLE       IsType 'com,feed_or,pos';

     HI_CLK_DISABLE       PIN 17;             "HI FERRO SCL
                                              "ENABLE
     HI_CLK_DISABLE       IsType 'com,feed_or,pos';

"Equivalences

     CK = .C.;

EQUATIONS

"Reset

     [Q1.RE,Q2.RE,Q3.RE] = COUNTER_CLR;     "RESET COUNTER
                                            "FROM EPLD3

     Q3 := COUNTER_EN&!Q3;                 "COUNTER ONLY COUNTS
                                          "IN STATES C,E,G,L,N.
                                           "SEE EPLD3.
```

```
      Q2 := (COUNTER_EN&Q2&!Q3)#
            (COUNTER_EN&!Q2&Q3);

      Q1 := (COUNTER_EN&Q1&!Q3)#
            (COUNTER_EN&Q1&!Q2&Q3)#
            (COUNTER_EN&!Q1&Q2&Q3);

      CNTR_7 = Q1&Q2&Q3;

      LO_CLK_DISABLE = !A0&(A#I#O#(H&!CLK)#
                              (J&!CLK)#(N&CNTR_7&!CLK));

      HI_CLK_DISABLE = !BHE&(A#I#O#(H&!CLK)#
                               (J&!CLK)#(N&CNTR_7&!CLK));

      STOP_START_SET = I#(O&!CLK);

TEST_VECTORS ([CLK,COUNTER_CLR,COUNTER_EN] ->
                                          [Q1,Q2,Q3,CNTR_7])
                  [0,1,0]   -> [0,0,0,0];
                  [CK,1,0]  -> [0,0,0,0];
                  [CK,0,1]  -> [0,0,1,0];
                  [CK,0,1]  -> [0,1,0,0];
                  [CK,0,1]  -> [0,1,1,0];
                  [CK,0,1]  -> [1,0,0,0];
                  [CK,0,1]  -> [1,0,1,0];
                  [CK,0,1]  -> [1,1,0,0];
                  [CK,0,1]  -> [1,1,1,1];
                  [CK,0,1]  -> [0,0,0,0];

TEST_VECTORS ([N,CNTR_7,A,I,H,J,O,CLK,A0,BHE] ->
              [HI_CLK_DISABLE,LO_CLK_DISABLE,STOP_START_SET])
              [1,1,0,0,0,0,0,1,0,1] -> [0,0,0];
              [1,0,0,0,0,0,0,0,0,1] -> [0,0,0];
              [1,0,0,0,0,0,0,1,0,1] -> [0,0,0];
              [0,0,1,0,0,0,0,0,0,1] -> [0,1,0];
              [0,0,1,0,0,0,0,1,0,1] -> [0,1,0];
              [0,0,0,1,0,0,0,0,0,1] -> [0,1,1];
              [0,0,0,1,0,0,0,1,0,1] -> [0,1,1];
              [0,0,0,0,1,0,0,0,0,1] -> [0,1,0];
              [0,0,0,0,1,0,0,1,0,1] -> [0,0,0];
              [0,0,0,0,0,1,0,0,0,1] -> [0,1,0];
              [0,0,0,0,0,1,0,1,0,1] -> [0,0,0];
              [0,0,0,0,0,0,1,0,0,1] -> [0,1,1];
              [0,0,0,0,0,0,1,1,0,1] -> [0,1,0];
              [1,1,0,0,0,0,0,1,1,0] -> [0,0,0];
              [1,0,0,0,0,0,0,0,1,0] -> [0,0,0];
              [1,0,0,0,0,0,0,1,1,0] -> [0,0,0];
              [0,0,1,0,0,0,0,0,1,0] -> [1,0,0];
              [0,0,1,0,0,0,0,1,1,0] -> [1,0,0];
              [0,0,0,1,0,0,0,0,1,0] -> [1,0,1];
```

```
                [0,0,0,1,0,0,0,1,1,0] -> [1,0,1];
                [0,0,0,0,1,0,0,0,1,0] -> [1,0,0];
                [0,0,0,0,1,0,0,1,1,0] -> [0,0,0];
                [0,0,0,0,0,1,0,0,1,0] -> [1,0,0];
                [0,0,0,0,0,1,0,1,1,0] -> [0,0,0];
                [0,0,0,0,0,0,1,0,1,0] -> [1,0,1];
                [0,0,0,0,0,0,1,1,1,0] -> [1,0,0];
END EPLD4
```

E. PROGRAM 'EPLD5.ABL'

```
MODULE EPLD5
FLAG '-r2'
TITLE 'FSM #1 -- FERRO I/O CONTROLLER 06/13/91'

     U1 DEVICE 'E0310';

"Input pins

     CLK                  PIN 1;                "100 KHZ CLOCK

     FERRO                PIN 2;                "CLEAR/ENABLE
                                                "FSM

     ACK                  PIN 3;                "MEMORY
                                                "ACKNOWLEDGE

     CNTR_7               PIN 4;                "FSM #2 = 7

     RW                   PIN 6;                " 1=RD, 0 = WR

     Q1,Q2,Q3,Q4          PIN 12,13,14,15;      "COUNTER
     Q1,Q2,Q3,Q4          IsType 'feed_reg,reg_d,pos';

     RTERM1               PIN 16;               "REDUCTION TERM
     RTERM1               IsType 'com,feed_or,pos';

"Equivalences

     X = .X.;
     C = .C.;
     COUNT = [Q1,Q2,Q3,Q4];

EQUATIONS


"Reset

     [Q1.RE,Q2.RE,Q3.RE,Q4.RE] = FERRO;         "RESET FSM

     RTERM1 = (!FERRO&CNTR_7&((COUNT == ^H2)#(COUNT == ^H6)#
                                              (COUNT == ^H4)))#
              (!FERRO&!ACK&(COUNT == ^H3))#
              (!FERRO&(COUNT == ^H0));
```

```
      Q4 := (!FERRO&!CNTR_7&(COUNT == ^HD))#
            (!FERRO&ACK&(COUNT == ^HC))#
            (!FERRO&!CNTR_7&(COUNT == ^HB))#
            (!FERRO&(COUNT == ^HA))#
            (!FERRO&!ACK&((COUNT == ^H5)#
            (COUNT ==^H3)#(COUNT==^H9)))#
            RTERM1;

      Q3 := (!FERRO&CNTR_7&Q1&Q2&!Q3&Q4)#
            (!FERRO&!CNTR_7&Q1&!Q2&Q3&Q4)#
            (!FERRO&!Q2&Q3&!Q4)#
            (!FERRO&!Q1&Q2&Q3&!Q4)#
            (!FERRO&!Q1&!Q2&!Q3&Q4)#
            (!FERRO&ACK&((!Q1&Q2&!Q3&Q4)#(Q1&!Q2&!Q3&Q4)))#
            (!FERRO&!ACK&!Q1&!Q2&Q3&Q4);

      Q2 := (!FERRO&Q1&Q2&!Q3)#
            (!FERRO&CNTR_7&Q1&!Q2&Q3&Q4)#
            (!FERRO&!Q1&Q2&Q3&!Q4)#
            (!FERRO&!Q1&Q2&!Q3&Q4)#
            (!FERRO&CNTR_7&!RW&!Q1&Q2&!Q3&!Q4)#
            (!FERRO&!CNTR_7&!Q1&Q2&!Q3&!Q4)#
            (!FERRO&ACK&!Q1&!Q2&Q3&Q4);

      Q1 := (!FERRO&Q1&Q2&!Q3)#
            (!FERRO&Q1&!Q2&Q3)#
            (!FERRO&Q1&!Q2&!Q3&Q4)#
            (!FERRO&!Q1&Q2&Q3&Q4)#
            (!FERRO&CNTR_7&RW&!Q1&Q2&!Q3&!Q4);

TEST_VECTORS ([CLK,FERRO,CNTR_7,RW,ACK]
                              ->[Q1,Q2,Q3,Q4])
                  [0,1,0,0,0]  -> [0,0,0,0];
                  [C,1,0,0,0]  -> [0,0,0,0];
                  [C,0,0,0,0]  -> [0,0,0,1];
                  [C,0,0,0,0]  -> [0,0,1,0];
                  [C,0,1,0,0]  -> [0,0,1,1];
                  [C,0,0,0,1]  -> [0,1,0,0];
                  [C,0,1,0,0]  -> [0,1,0,1];
                  [C,0,0,0,1]  -> [0,1,1,0];
                  [C,0,1,0,0]  -> [0,1,1,1];
                  [C,0,0,0,0]  -> [1,0,0,0];
                  [C,0,0,0,0]  -> [0,0,0,0];
                  [C,0,0,0,0]  -> [0,0,0,1];
                  [C,0,0,0,0]  -> [0,0,1,0];
                  [C,0,1,0,0]  -> [0,0,1,1];
                  [C,0,0,0,1]  -> [0,1,0,0];
                  [C,0,1,1,0]  -> [1,0,0,1];
                  [C,0,0,0,1]  -> [1,0,1,0];
                  [C,0,0,0,0]  -> [1,0,1,1];
                  [C,0,1,0,0]  -> [1,1,0,0];
```

```
                [C,0,0,0,1]  -> [1,1,0,1];
                [C,0,1,0,0]  -> [1,1,1,0];
                [C,0,0,0,0]  -> [0,0,0,0];

END EPLD5
```

F. PROGRAM 'EPLD6.ABL'

```
MODULE EPLD6
TITLE 'FSM #1 -- OUTPUT STATE DECODE 07/01/91'

     U1 DEVICE 'E0310';

"Input pins

     Q1,Q2,Q3,Q4          PIN 2,3,4,5;            "FSM COUNTER

"Output pins

     A,B,C,D,E,F        PIN 12,13,14,15,16,17; "OUTPUT STATES
     A,B,C,D,E,F          IsType 'com,feed_or,pos';

     G,H                 PIN 18,19;             "OUTPUT STATES
     G,H                   IsType 'com,feed_or,pos';

EQUATIONS

     A = (!Q1&!Q2&!Q3&!Q4);

     B = (!Q1&!Q2&!Q3&Q4);

     C = (!Q1&!Q2&Q3&!Q4);

     D = (!Q1&!Q2&Q3&Q4);

     E = (!Q1&Q2&!Q3&!Q4);

     F = (!Q1&Q2&!Q3&Q4);

     G = (!Q1&Q2&Q3&!Q4);

     H = (!Q1&Q2&Q3&Q4);

TEST_VECTORS ([Q1,Q2,Q3,Q4] ->[A,B,C,D,E,F,G,H])
                  [0,0,0,0]  -> [1,0,0,0,0,0,0,0];
                  [0,0,0,1]  -> [0,1,0,0,0,0,0,0];
                  [0,0,1,0]  -> [0,0,1,0,0,0,0,0];
                  [0,0,1,1]  -> [0,0,0,1,0,0,0,0];
                  [0,1,0,0]  -> [0,0,0,0,1,0,0,0];
                  [0,1,0,1]  -> [0,0,0,0,0,1,0,0];
                  [0,1,1,0]  -> [0,0,0,0,0,0,1,0];
                  [0,1,1,1]  -> [0,0,0,0,0,0,0,1];

END EPLD6
```

G. PROGRAM 'EPLD7.ABL'

```
MODULE EPLD7
TITLE 'FSM #1 -- OUTPUT STATE DECODE 07/31/91'

     U1 DEVICE 'E0310';

"Input pins

     Q1,Q2,Q3,Q4          PIN 2,3,4,5;            "COUNTER

"Output pins

     I,J,K,L,M,N,O        PIN 12,13,14,15,16,17,18; "OUTPUT
                                                    "STATES
     I,J,K,L,M,N,O        IsType 'com,feed_or,pos';

EQUATIONS

     I = (Q1&!Q2&!Q3&!Q4);

     J = (Q1&!Q2&!Q3&Q4);

     K = (Q1&!Q2&Q3&!Q4);

     L = (Q1&!Q2&Q3&Q4);

     M = (Q1&Q2&!Q3&!Q4);

     N = (Q1&Q2&!Q3&Q4);

     O = (Q1&Q2&Q3&!Q4);

TEST_VECTORS ([Q1,Q2,Q3,Q4] ->[I,J,K,L,M,N,O])
                  [1,0,0,0]  -> [1,0,0,0,0,0,0];
                  [1,0,0,1]  -> [0,1,0,0,0,0,0];
                  [1,0,1,0]  -> [0,0,1,0,0,0,0];
                  [1,0,1,1]  -> [0,0,0,1,0,0,0];
                  [1,1,0,0]  -> [0,0,0,0,1,0,0];
                  [1,1,0,1]  -> [0,0,0,0,0,1,0];
                  [1,1,1,0]  -> [0,0,0,0,0,0,1];

END EPLD7
```

H. PROGRAM 'EPLD8.ABL'

```
MODULE EPLD8
TITLE 'FSM #1 -- OUTPUT STATE DECODE 07/01/91'

     U1 DEVICE 'E0310';

"Input pins

     Q1,Q2,Q3,Q4          PIN 2,3,4,5;          "COUNTER

"Output pins

     MUXA,MUXB            PIN 13,14;            "MUX CONTROL
                                                " LINES
     MUXA,MUXB            IsType 'com,feed_or,pos';

     S0,S1                PIN 15,16;           "SHIFT REG MODE
                                                "CONTROL
     S0,S1                IsType 'com,feed_or,pos';

EQUATIONS

     MUXA = (Q1&!Q2&Q3&!Q4)#
            (Q1&!Q2&!Q3&Q4)#
            (!Q1&!Q2&Q3&Q4);

     MUXB = (Q1&!Q2&Q3&!Q4)#
            (Q1&!Q2&!Q3&Q4)#
            (!Q1&Q2&!Q3&Q4);

     S1 = (Q1&!Q2&Q3&!Q4)#
          (!Q2&!Q3&Q4)#
          (!Q1&Q2&!Q3&Q4)#
          (!Q1&!Q2&Q3&Q4);

     S0 = (Q1&Q2&!Q3)#
          (!Q2&Q3)#
          (!Q2&!Q3&Q4)#
          (!Q1&Q2&!Q4)#
          (!Q1&Q2&!Q3&Q4);

TEST_VECTORS ([Q1,Q2,Q3,Q4] ->[MUXA,MUXB,S0,S1])
                  [0,0,0,0]  -> [0,0,0,0];
                  [0,0,0,0]  -> [0,0,0,0];
                  [0,0,0,1]  -> [0,0,1,1];
                  [0,0,1,0]  -> [0,0,1,0];
                  [0,0,1,1]  -> [1,0,1,1];
                  [0,1,0,0]  -> [0,0,1,0];
                  [0,1,0,1]  -> [0,1,1,1];
```

```
                    [0,1,1,0]  -> [0,0,1,0];
                    [0,1,1,1]  -> [0,0,0,0];
                    [1,0,0,0]  -> [0,0,0,0];
                    [1,0,0,1]  -> [1,1,1,1];
                    [1,0,1,0]  -> [1,1,1,1];
                    [1,0,1,1]  -> [0,0,1,0];
                    [1,1,0,0]  -> [0,0,1,0];
                    [1,1,0,1]  -> [0,0,1,0];
                    [1,1,1,0]  -> [0,0,0,0];

END EPLD8
```

I. PROGRAM 'EPLD9.ABL'

```
MODULE EPLD9
TITLE 'ACKNOWLEDGE ENABLE 08/12/91'

     U1 DEVICE 'E0310';

"Input pins

     CLK                  PIN 1;              "100 KHZ CLK
     CNTR_7               PIN 2;              "SHIFT COUNT = 7
     O,C,E,G,L            PIN 3,4,5,6,7;      "FSM #1 STATES
     B,H,I,K              PIN 8,9,11,12;      "FSM #1 STATES
     BHE,A0              PIN 13,14;         "8086 CONTROL LINES


"Output pins

     HI_START_STOP_EN     PIN 18;          "SDA START OR STOP
                                           "CONDITION BUS ENABLE
     HI_START_STOP_EN     IsType 'com,feed_or,pos';

     LO_START_STOP_EN     PIN 17;          "SDA START OR STOP
                                           "CONDITION BUS ENABLE
     LO_START_STOP_EN     IsType 'com,feed_or,pos';

     ACK_EN              PIN 19;            "SDA BUS ACK ENABLE
     ACK_EN               IsType 'com,feed_or,pos';

EQUATIONS

     ACK_EN = (C&CNTR_7&!CLK)#
              (E&CNTR_7&!CLK)#
              (G&CNTR_7&!CLK)#
              (L&CNTR_7&!CLK);

     HI_START_STOP_EN = !BHE&(I#
                        (H&!CLK)#
                        (K&CLK)#
                        (B&CLK)#O);

     LO_START_STOP_EN = !A0&(I#
                        (H&!CLK)#
                        (K&CLK)#
                        (B&CLK)#O);

TEST_VECTORS ([C,CNTR_7,!CLK,E,G,L] -> [ACK_EN])
              [0,1,1,0,0,0] -> [0];
              [1,1,1,0,0,0] -> [1];
              [0,1,1,0,0,1] -> [1];
```

```
                [0,1,1,0,1,0] -> [1];
                [0,1,1,1,0,0] -> [1];
                [1,0,1,0,0,0] -> [0];
                [0,0,1,0,0,1] -> [0];
                [0,0,1,0,1,0] -> [0];
                [0,0,1,1,0,0] -> [0];

TEST_VECTORS ([A0,BHE,B,I,H,K,O,CLK] ->
                    [HI_START_STOP_EN,LO_START_STOP_EN])
                [0,1,1,0,0,0,0,0] -> [0,0];
                [0,1,1,0,0,0,0,1] -> [0,1];
                [0,1,0,1,0,0,0,1] -> [0,1];
                [0,1,0,1,0,0,0,0] -> [0,1];
                [0,1,0,0,1,0,0,0] -> [0,1];
                [0,1,0,0,1,0,0,1] -> [0,0];
                [0,1,0,0,0,1,0,1] -> [0,1];
                [0,1,0,0,0,1,0,0] -> [0,0];
                [0,1,0,0,0,0,1,1] -> [0,1];
                [0,1,0,0,0,0,1,0] -> [0,1];
                [1,0,1,0,0,0,0,0] -> [0,0];
                [1,0,1,0,0,0,0,1] -> [1,0];
                [1,0,0,1,0,0,0,1] -> [1,0];
                [1,0,0,1,0,0,0,0] -> [1,0];
                [1,0,0,0,1,0,0,0] -> [1,0];
                [1,0,0,0,1,0,0,1] -> [0,0];
                [1,0,0,0,0,1,0,1] -> [1,0];
                [1,0,0,0,0,1,0,0] -> [0,0];
                [1,0,0,0,0,0,1,1] -> [1,0];
                [1,0,0,0,0,0,1,0] -> [1,0];

END EPLD9
```

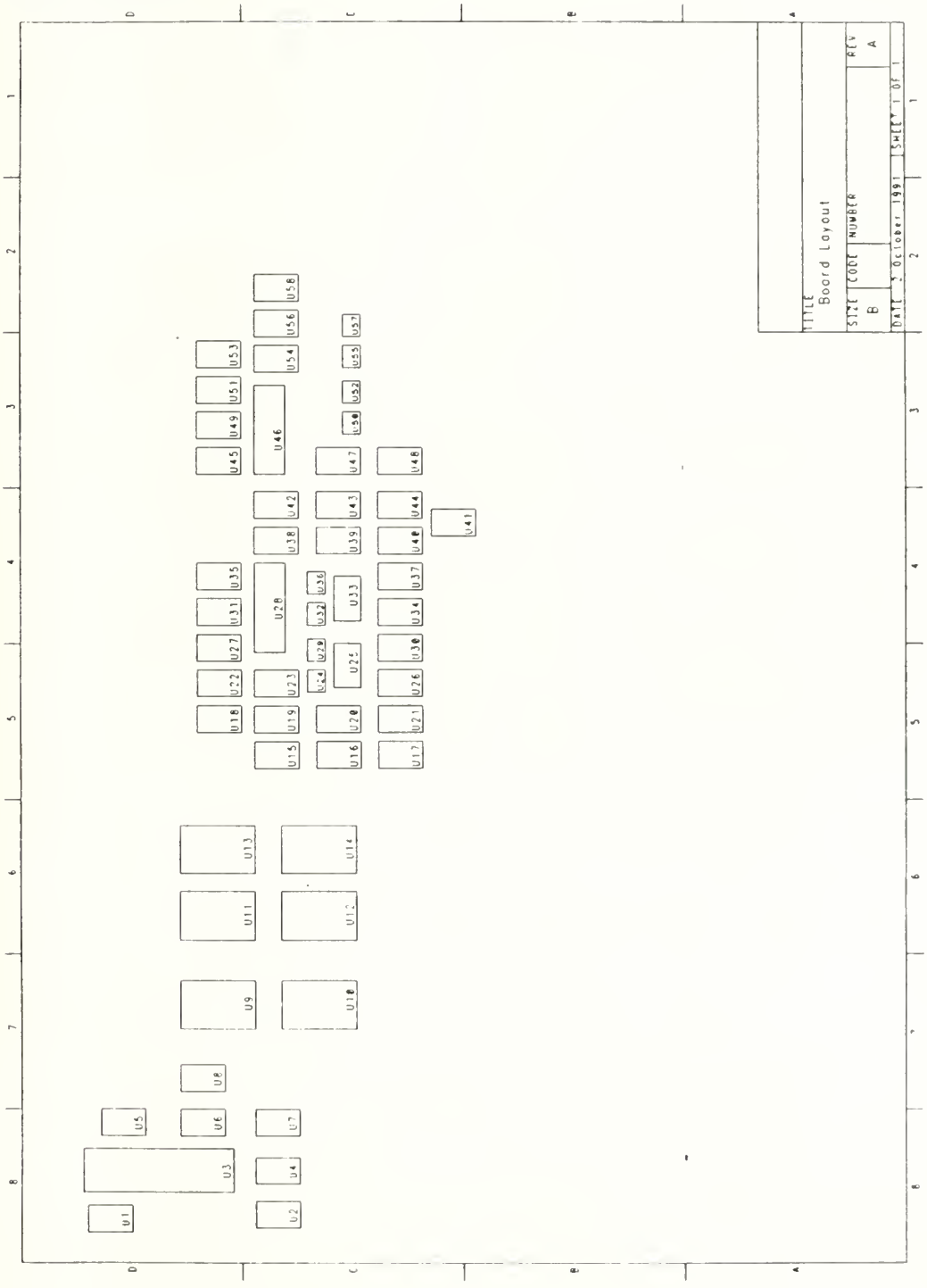
J. PROGRAM 'EPLD10.ABL'

```
MODULE EPLD10
FLAG '-r2'
TITLE 'MEMORY CHIP SELECT 09/10/91'

     U1 DEVICE 'E0310';

"Input pins

     A0,BHE              PIN 2,3;        "8086 CONTROL SIGNALS
     A14,A13,A12          PIN 4,5,6;       "ADDRESS LINES
     I,O                  PIN 7,8;         "FERRO FSM STATES

"Output pins

     LOWRAM1             PIN 12;   "LOW BYTE FIRST RAM BANK CS
     LOWRAM1              IsType 'com,feed_or,neg';

     LOWRAM2              PIN 13;   "LOW BYTE 2ND RAM BANK CS
     LOWRAM2              IsType 'com,feed_or,neg';

     HIRAM1              PIN 14;   "HI BYTE FIRST RAM BANK CS
     HIRAM1               IsType 'com,feed_or,neg';

     HIRAM2               PIN 15;   "HI BYTE 2ND RAM BANK CS
     HIRAM2               IsType 'com,feed_or,neg';

     ROMCS                PIN 16;   "ROM CHIP SELECT
     ROMCS                IsType 'com,feed_or,neg';

     FERRO                PIN 17;   "FERRO MEMORY CHIP SELECT
     FERRO                IsType 'com,feed_or,neg';

     RDY1                PIN 18;     "WAIT STATE GENERATOR FOR
                                       "8086
     RDY1                 IsType 'com,feed_or,pos';

"Equivalences

     X = .X.;
     ADDRESS =[A14,A13, A12,X,X,X, X,X,X,X, X,X,X,X];

EQUATIONS

    !LOWRAM1 = !A0&(ADDRESS >= ^H2000)&(ADDRESS< ^H2800);

    !LOWRAM2 = !A0&(ADDRESS >= ^H2800)&(ADDRESS<= ^H2FFF);

    !HIRAM1  = !BHE&(ADDRESS >= ^H2000)&(ADDRESS< ^H2800);
```

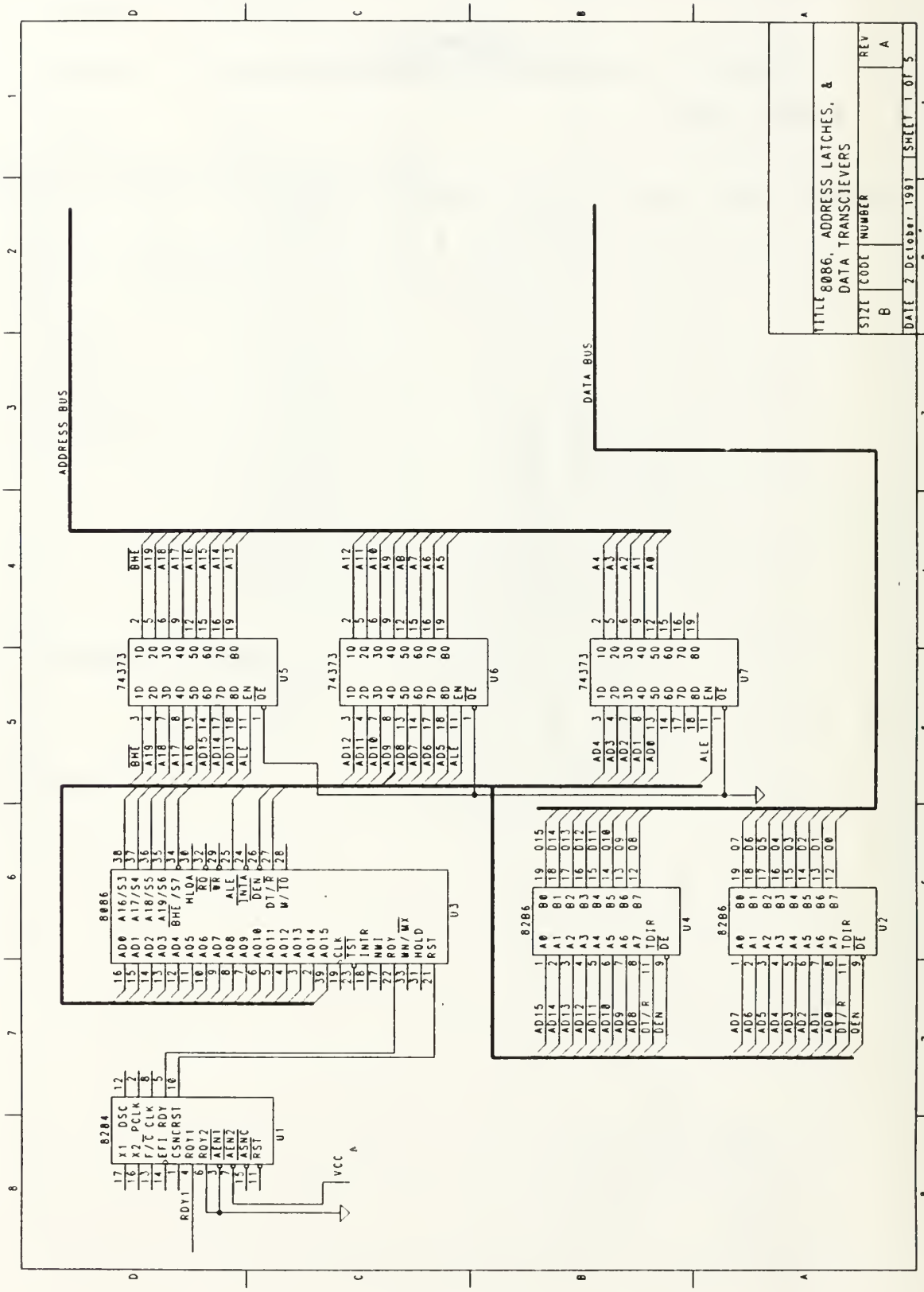
```
    !HIRAM2   = !BHE&(ADDRESS >= ^H2800)&(ADDRESS<= ^H2FFF);

    !ROMCS    = (ADDRESS >= ^H3000);

    !FERRO    = (ADDRESS >= ^H1000)&(ADDRESS < ^H1800);

    RDY1      = FERRO # I # O;

TEST_VECTORS ([ADDRESS,A0,BHE]
               ->[LOWRAM1,LOWRAM2,HIRAM1,HIRAM2,ROMCS,FERRO])
                [^H2000,0,1]  -> [0,1,1,1,1,1];
                [^H2000,1,0]  -> [1,1,0,1,1,1];
                [^H27FF,0,1]  -> [0,1,1,1,1,1];
                [^H27FF,1,0]  -> [1,1,0,1,1,1];
                [^H2800,0,1]  -> [1,0,1,1,1,1];
                [^H2800,1,0]  -> [1,1,1,0,1,1];
                [^H2FFF,0,1]  -> [1,0,1,1,1,1];
                [^H2FFF,1,0]  -> [1,1,1,0,1,1];
                [^H3000,0,1]  -> [1,1,1,1,0,1];
                [^H3000,1,0]  -> [1,1,1,1,0,1];
                [^H1000,0,1]  -> [1,1,1,1,1,0];
                [^H1000,1,0]  -> [1,1,1,1,1,0];
                [^H17FF,0,1]  -> [1,1,1,1,1,0];
                [^H17FF,1,0]  -> [1,1,1,1,1,0];
                [^H1800,1,0]  -> [1,1,1,1,1,1];
                [^H1800,0,1]  -> [1,1,1,1,1,1];

TEST_VECTORS ([FERRO,I,O] -> [RDY1])
              [1,0,0]     -> [1];
              [0,1,0]     -> [1];
              [0,0,1]     -> [1];

END EPLD10
```

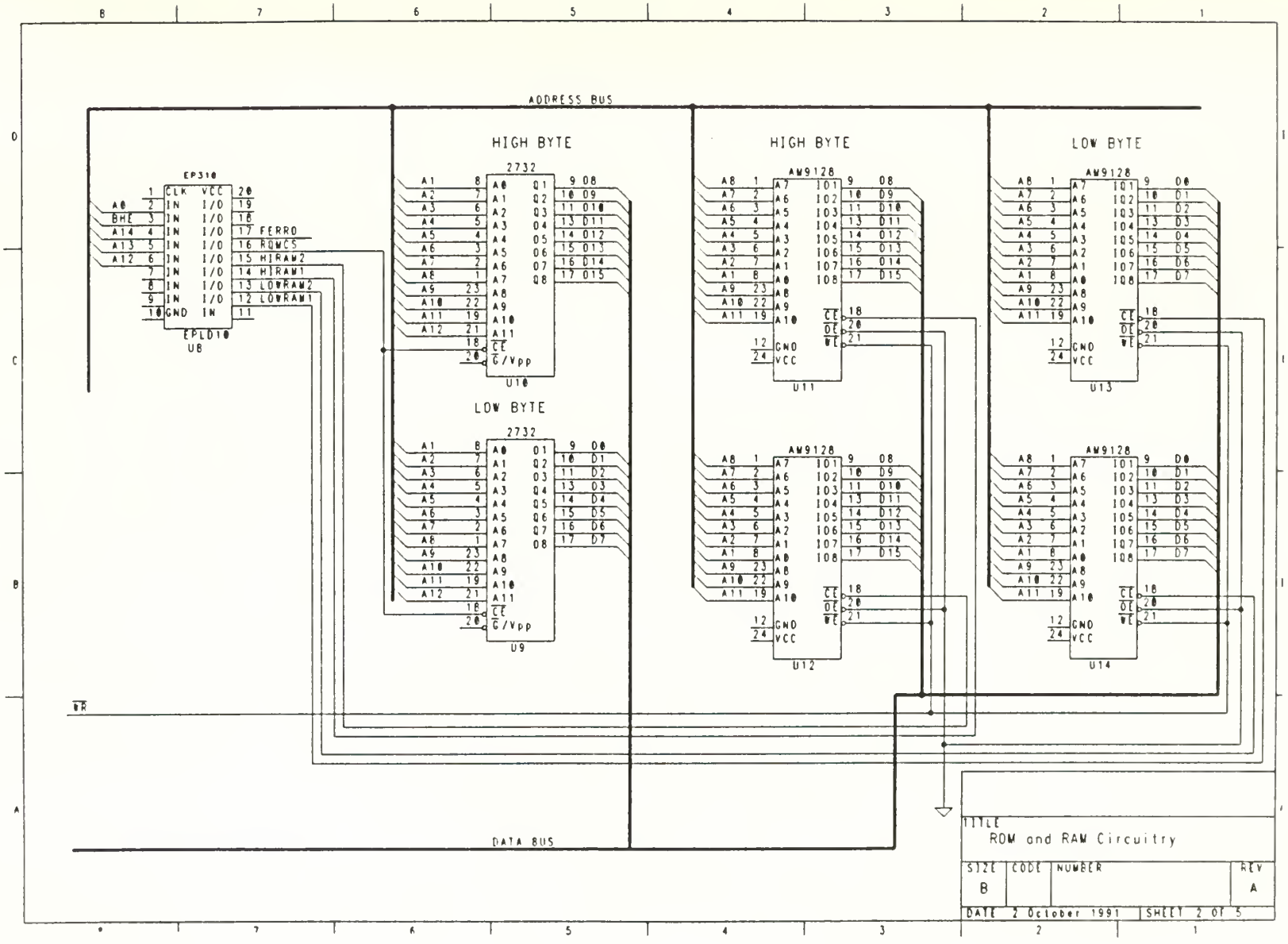
APPENDIX C
CIRCUIT SCHEMATICS

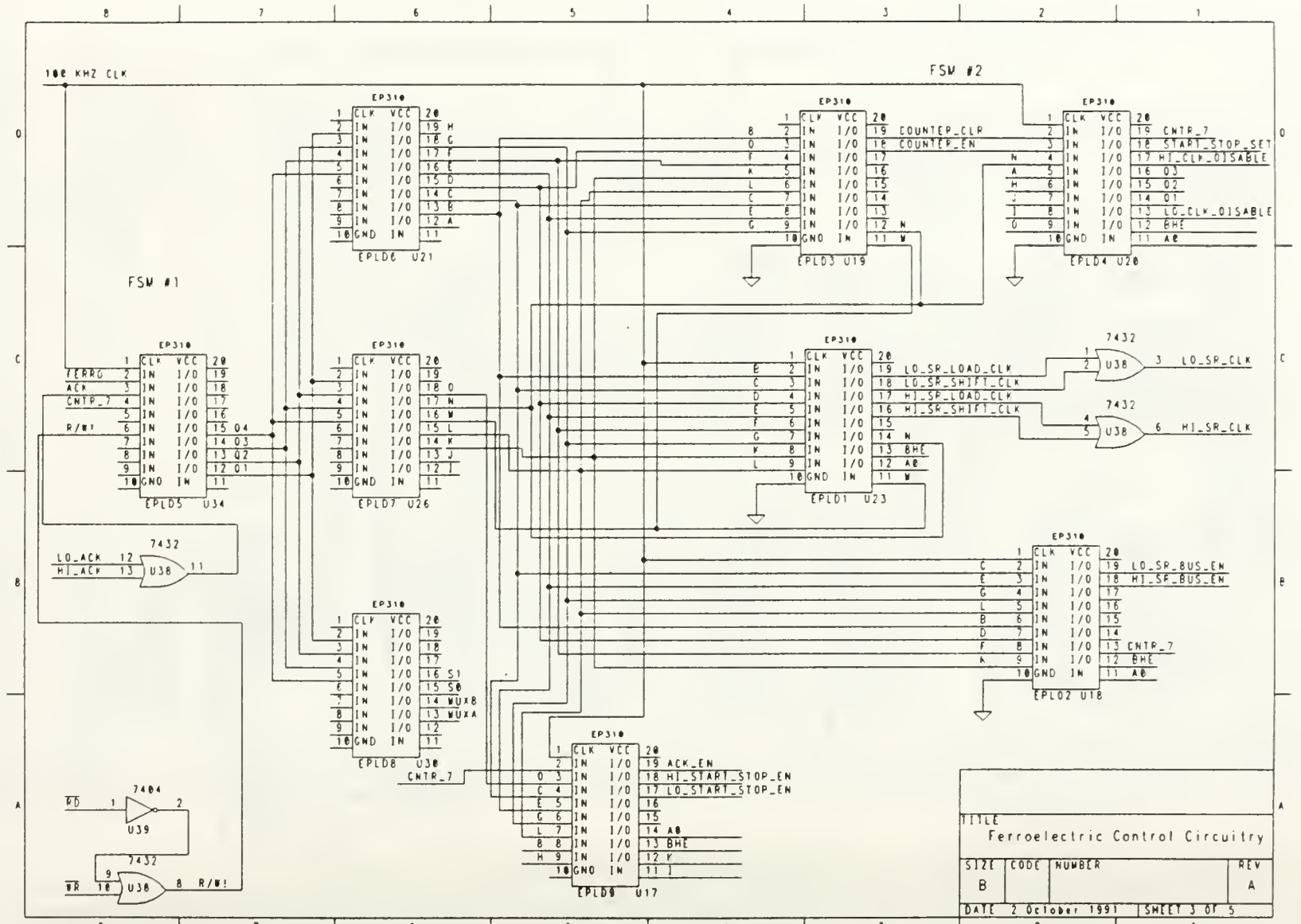
A. BOARD LAYOUT

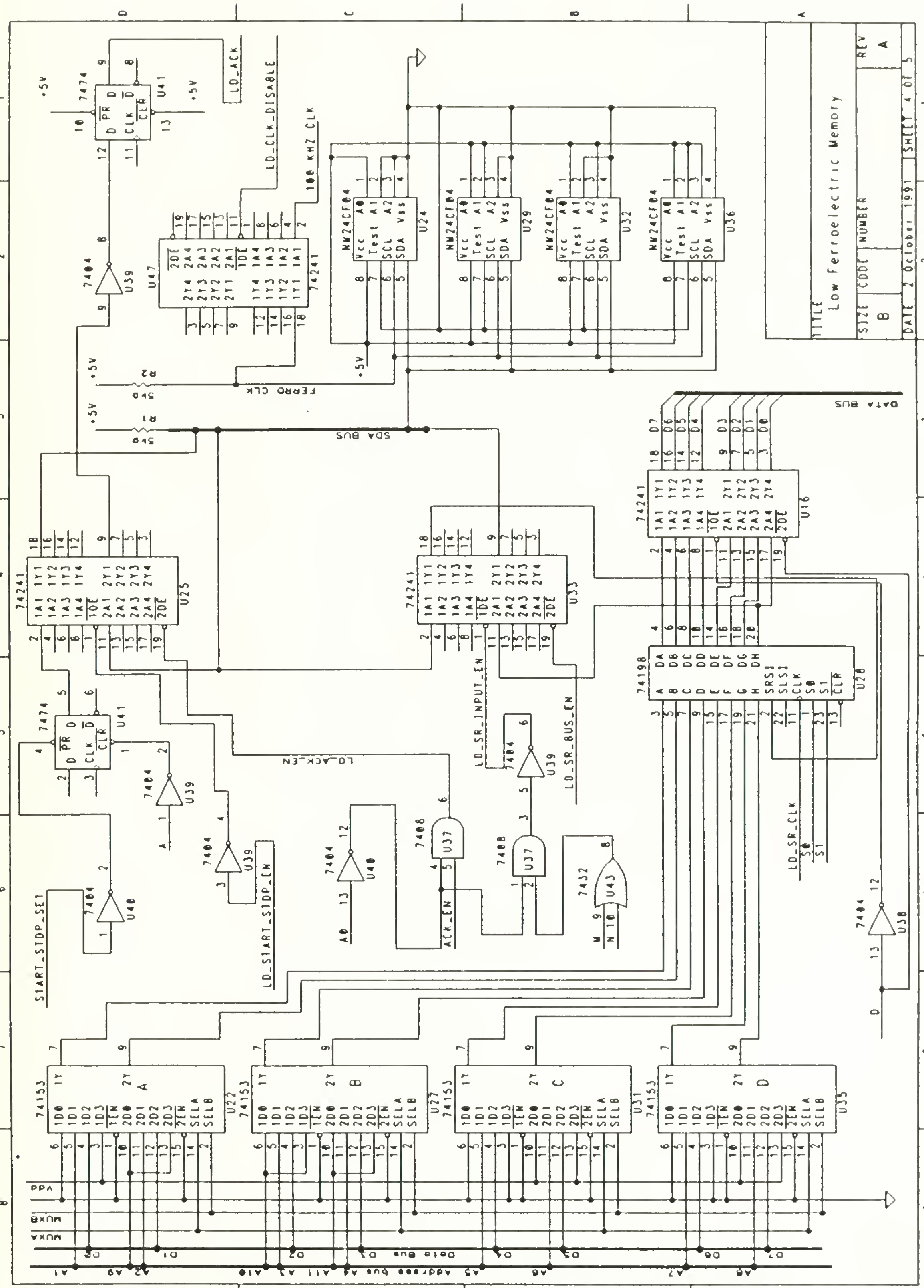


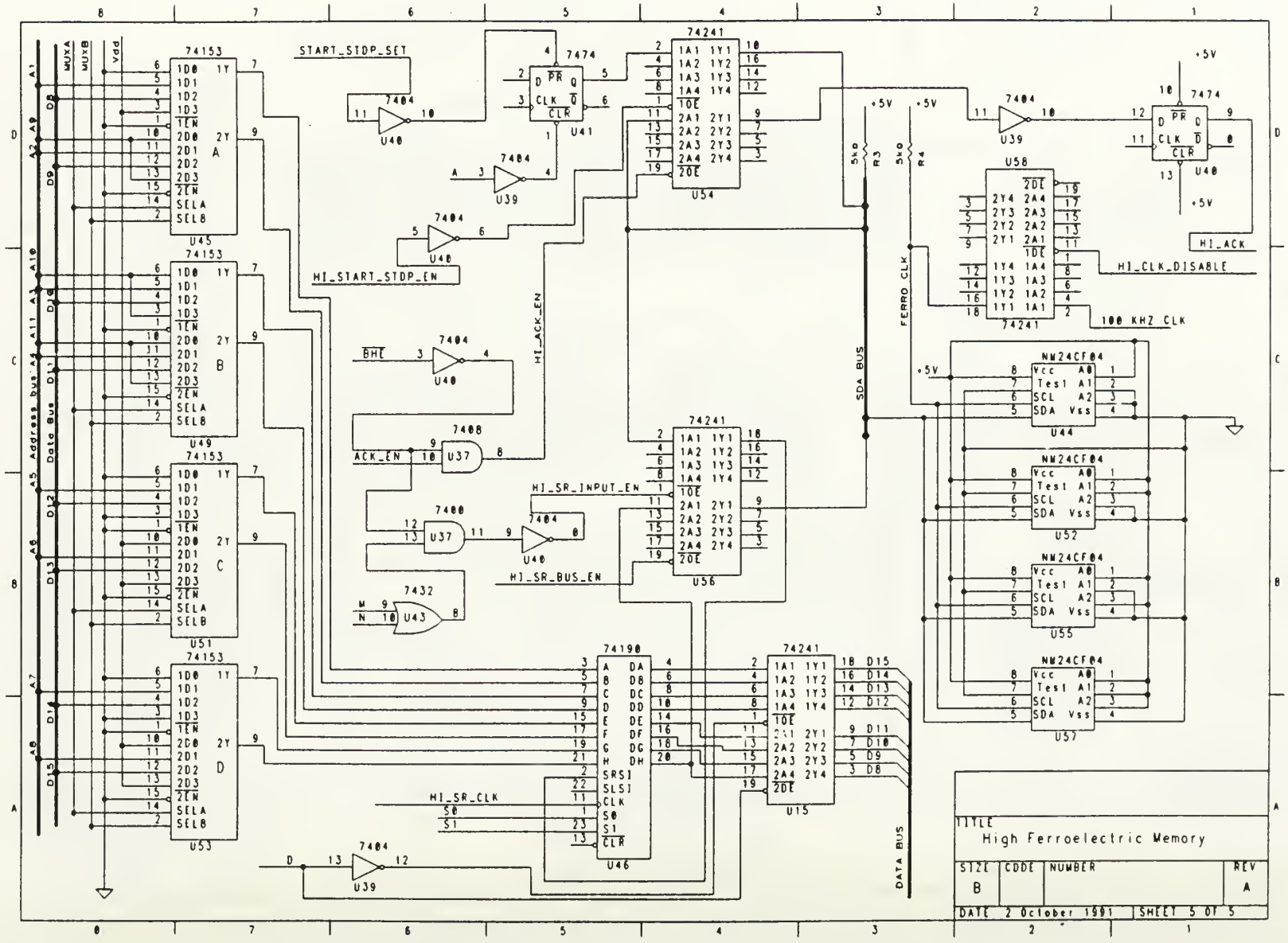
B. SCHEMATICS




ADDRESS BUS
HIGH BYTE
LOW BYTE
HIGH BYTE
LOW BYTE
EP310
EPLD10
U8
2732
U10
2732
U9
AM9128
U11
AM9128
U12
AM9128
U13
AM9128
U14
FERRO
ROMCS
HIRAM2
HIRAM1
LOWRAM2
LOWRAM1
DATA BUS
TITLE
ROM and RAM Circuitry
SIZE
B
CODE
NUMBER
REV
A
DATE 2 October 1991
SHEET 2 OF 5


100 KHZ CLK
FSM #1
FSM #2
EPLD6 U21
EPLD3 U19
EPLD4 U20
EPLD5 U34
EPLD7 U26
EPLD1 U23
EPLD2 U18
EPLD8 U30
EPLD0 U17
7432
7404
U38
U39
LO_SR_CLK
HI_SR_CLK
LO_ACK
HI_ACK
CNTR_7
R/W!
COUNTER_CLR
COUNTER_EN
LO_SR_LOAD_CLK
LO_SR_SHIFT_CLK
HI_SR_LOAD_CLK
HI_SR_SHIFT_CLK
LO_SR_BUS_EN
HI_SR_BUS_EN
ACK_EN
HI_START_STOP_EN
LO_START_STOP_EN
TITLE
Ferroelectric Control Circuitry
SIZE
B
CODE
NUMBER
REV
A
DATE 2 October 1991
SHEET 3 OF 5


TITLE
Low Ferroelectric Memory
SIZE B
CODE
NUMBER
REV A
DATE 2 October 1991
SHEET 4 OF 5


74153
U45
U49
U51
U53
74241
U54
U56
U15
74190
U46
7474
U41
7404
U40
U39
7408
U37
7432
U43
START_STDP_SET
HI_START_STDP_EN
HI_ACK_EN
BHE
ACK_EN
HI_SR_INPUT_EN
HI_SR_BUS_EN
HI_SR_CLK
SDA BUS
DATA BUS
FERRO CLK
+5V
5kΩ
R3
R4
U58
HI_ACK
HI_CLK_DISABLE
100 KHZ CLK
NM24CF04
U44
U52
U55
U57
TITLE
High Ferroelectric Memory
SIZE
B
CODE
NUMBER
REV
A
DATE 2 October 1991
SHEET 5 OF 5

LIST OF REFERENCES

1. Bondurant, D., and Gnadinger, F., "Ferroelectrics for Nonvolatile RAMs," *Spectrum* v. 26, pp. 30-33, July 1989.

2. Bogert, H. D., "FERRAM: The Memory the Market Always Wanted", *Dataquest Research Newsletter*, pp. 1-12, January 1988.

3. Intel Corporation, *The 8086 Family User's Manual*, October 1979.

4. Intel Corporation, *8086/8088 User's Manual Programmer's and Hardware Reference*, 1989.

5. National Semiconductor, *NM24CF04, 4096-Bit (512x8) CMOS Serial Nonvolatile Memory*, November 1990.

BIBLIOGRAPHY

1. Evans, J. T., Womack, R., and Tolsch, D., "Ferroelectric Non-volatile Memory," *Proceedings of the IEEE National Aerospace and Electronics Conference, v. 1*, pp. 65-72, 1988.

2. Scott, J.F., and Paz de Araujo, C. A., "Ferroelectric Memories," *Science, v. 246*, pp. 1400-1405, 15 December 1989.

3. Simons, M., "Radiation effects in GaAs integrated circuits: A comparison with silicon," *Proceedings of the IEEE Gallium Arsenide Integrated Circuit Symposium*, pp. 124 - 128, 1983.

4. Harris Semiconductor, *Rad-Hard/Hi-Rel Data Book*, 1990.

5. Bogard, M., "Ferroelectric Memory Applications," *Dataquest Report*, pp 1-12, 1988.

6. Scott, J. F. and others, "Radiation Effects on Ferroelectric Thin-Film Memories: Retention Failure Mechanisms", *Journal of Applied Physics*, v. 66, 1 August 1989.

7. Bondurant, D., "Ferroelectric RAM Memory Family for Critical Data Storage", *Proceedings of the First Symposium on Integrated Ferroelectrics*, Colorado Springs, CO, pp. 212-215, March 1989.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Department Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5002	1
4.	Commandant of the Marine Corps Code TE-06 Headquarters, U.S. Marine Corps Washington, D.C. 20380-0001	1
5.	Professor Douglas Fouts, Code EC/Fs Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5002	3
6.	Professor Herschel Loomis, Code EC/Lm Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5002	1
7.	CAPT Thomas C. Gonter MCRDAC C2IS Quantico, Virginia 22134-5080	2

Thesis
G54753 Gonter
c.1 A microprocessor interface for the NM24CF04 serial-access ferroelectric memory.