

DATA TRANSFER OPERATIONS IN A  
68008 BASED MICROCOMPUTER  
SYSTEM

BY

ENEFAA GEORGE DOUGLAS

Bachelor of Science

Oklahoma State University

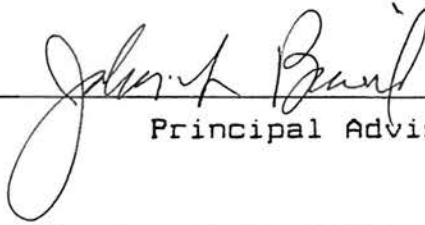
Stillwater, Oklahoma

1983

Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
May, 1985

DATA TRANSFER OPERATIONS IN A 68008  
BASED MICROCOMPUTER SYSTEM

APPROVED:

A handwritten signature in cursive script, appearing to read "John A. Beaulieu", is written over a horizontal line.

Principal Adviser

## PREFACE

Understanding the methods of data transfer in a computer system is essential to its maintenance. Different microprocessors transfer data differently. Data is usually transferred between the processor and the system memory or between peripherals and the processor.

Three methods of data transfer were examined in this report namely, synchronous data transfer, asynchronous data transfer and a combination of the above two methods.

Synchronous data transfer uses the clock to connect sequence and time. Intervals between clock transitions must therefore be such as to permit enough time for the activities planned for that interval. Asynchronous data transfer on the other hand is independent of clock frequency at the system level. Communication on the system bus is without a timing signal allowing any amount of time between data bytes or words. To indicate when data has been received or sent, the external device sends an acknowledgement signal to the processor.

Synchronous/Asynchronous mode of operation allows the synchronous device to be clocked at its maximum operating frequency using an externally generated enable (E) clock. This way, the processor runs asynchronously and the acknowledgement signal is generated synchronously with the external enable clock.

No one method is superior but the last method is

recommended. The method to use depends on the application.

I wish to express my gratitude to all the people who assisted me in this work. In particular, I would like to thank Dr. Neal Willison for his assistance with special technical background materials, Dr. John Baird for his patience and assistance, Dr. Craig Anderson for serving on the committee, Dr. Larry Jones for allowing me to use his texts for references and Miss Jane Bushaw for her support.

## TABLE OF CONTENT

Chapter		Page
I	INTRODUCTION . . . . .	1
	Statement of problem . . . . .	1
	Objective . . . . .	2
II	SYSTEM DESCRIPTION . . . . .	3
	Criteria for choice of hardware . . . . .	3
	Hardware description of processor . . . . .	4
	Data transfer methods and implementation . . . . .	8
III	DATA TRANSFER METHODS COMPARISON AND CONCLUSION . . . . .	21
	Asynchronous versus synchronous methods . . . . .	21
	Conclusion . . . . .	23
	SELECTED BIBLIOGRAPHY . . . . .	27
	APPENDIXES . . . . .	29
	APPENDIX A - Glossary of terms . . . . .	29

Chapter	Page
APPENDIX B - System Block Diagram and Programming model . . . . .	34
APPENDIX C - Memory DTACK Generation. . . . .	37
APPENDIX D - System Cycle Flowcharts . . . . .	40
APPENDIX E - System Circuit and Interfacing Logic . . . . .	46
APPENDIX F - System Timing And Bus Cycle Operations . . . . .	49
APPENDIX G - Input/Output Decoding . . . . .	52

## LIST OF FIGURES

Figure		Page
1.	System Block Diagram . . . . .	35
2.	Processor Programming Model . . . . .	36
3.	ROM/SRAM DTACK Generator . . . . .	38
4.	DRAM DTACK Generator . . . . .	39
5.1.	Read-Modify-Write Cycle Flowchart . . . . .	41
5.2.	Byte Read Cycle Flowchart . . . . .	42
5.3.	Byte Write Cycle Flowchart . . . . .	43
5.4.	Vector Acquisition Flowchart . . . . .	44
6.1.	Synchronous Device Cycle Flowchart . . . . .	45
6.2.	Synchronous Circuit Operation and Interfacing . . . . .	47
7.	Synchronous / Asynchronous Interface Logic . . . . .	48
8.	MC68008 and 6800-type Bus Timing for Read Operation . . . . .	50
9.	Read Cycle Timing Diagram for Mc68008 . . . . .	51
10.	Input / Output Decoder . . . . .	53

## CHAPTER ONE

### INTRODUCTION

One of the exciting features of the computer field is that concepts and applications once considered exotic and unattainable can become commonplace almost overnight. This rapid rate of change places a special burden on computer professionals if they are to stay up-to-date with the latest developments. The best insurance against becoming obsolete is a solid educational background that lays a foundation for lifelong learning. A major responsibility of the computer educator is to make sure that the educational program properly prepare students for entrance into the computer science and engineering profession.

For computer educators, developing and maintaining an up-to-date curriculum is a major problem. However, these problems are also important to the industrial community, since the educational background of recent graduates determines how well they will be able to contribute to future computer developments. The computer society is very fortunate to have a group of members from academia, government, and industry who, through the participation of



all parties, design a wide range of projects to improve the educational processes at all levels.

Closely related to the educational programs in computer science and engineering are the programs in computer technology. These programs provide graduates with the understanding and technical background needed to maintain both the hardware and software of modern computer systems. Unfortunately, many established programs in computer technology lack the depth required to prepare a student with the tools needed to solve real-world problems.

The main objective of this report therefore is to devise a step by step procedure in designing for a synchronous/asynchronous data transfer operation in a 68008 based microcomputer system.

The devised steps to follow are:

- 1) choosing which processor to use,
- 2) the processor's hardware description,
- 3) asynchronous data transfer operations,
- 4) synchronous data transfer operations and
- 5) synchronous/asynchronous data transfer operations.

## CHAPTER TWO

### SYSTEM DESCRIPTION

#### CHOOSING THE PROCESSOR

As more and more 16-bit microprocessors appear on the market, the question of which one to base an educational system on becomes more difficult. Ideally, most designers will base their decision on thorough hands-on experience, but getting hold of an inexpensive system is almost impossible.

The System proposed here is based on an 8/16/32 bit processor ( 8-bit external data bus; 8/16/32-bit internal data bus ). It is the Motorola MC68008 microprocessor.

There are several reasons for choosing this processor, namely:

a) The 8-bit external data bus is readily interfaced to already existing 8-bit processors, memories, and peripheral chips.

b) The 68008 is source and object code compatible with the 16-bit 68000 processor already used in such

microcomputers as the Apple Macintosh, the Altos, the Commodore Amiga, the Atari 520ST, and the Radio Shack IRS-80 Models 16 and 6000.

c) The 68008 has a linear address space (ie, a non-segmented address space) of one megabyte as compared to the 64 kilobytes of standard 8-bit processors.

d) The 68008 like all the other of the 68000 family of processors have powerful addressing modes designed to implement high level languages.

e) Proven reliability and commitment on the part of Motorola to maintain compatibility between its new chips and the 68000 processor.

f) Memory mapped input/output (I/O). This I/O scheme reduces component count (Dougherty, 1986).

#### Hardware Description Of The 68008 Microprocessor

The main component of any computer system is the Central Processing Unit (CPU). It is the duty of the processor to provide or request data and select proper address for this data. In addition, the CPU is capable of performing a limited set of mathematical and logical operations on the data.

Proper design can reduce the total chip count in a microcomputer system without increasing cost, limiting expandability, or sacrificing performance. These goals were

accomplished by using the MC68008 microprocessor in combination with four other large scale integrated (LSI) chips and some support chips as shown in the block diagram of Figure 1.

The MC68008 has the same internal architecture as, and is fully software compatible with the MC68000 microprocessor unit (MPU), but has an eight bit external data bus. It therefore allows the design of a cost-effective system using an eight bit non-multiplexed data bus. It also provide the benefits of a 32-bit microprocessor architecture.

The 68008 is available in two packages ( a 48 DIP and a 52 DIP packages). The 48 pin package was used for this design.

The large non-segmented address space of the 68008 allows large modular programs to be developed and executed efficiently. This means the program segment sizes are to be determined by the application rather than the designer who may otherwise be forced to adopt an arbitrary segment size without regard to the application's individual requirements.

The programmer's model of the 68008 processor is identical to that of the 68000 as shown in Figure 2. It consists of seventeen 32-bit registers, a 32-bit program counter, and a 16-bit status register. The first eight registers (D0-D7) are used as data registers for byte (8-bit), word (16-bit), and long word (32-bit) operations. The second set of seven registers (A0-A6), the user stack pointer (A7), and the system stack pointer (A7') may be used

as software stack pointers and base address registers. The seventeen registers can also be used as general purpose 32-bit registers giving the programmer additional flexibility.

#### Program Privilege Scheme Used By The 68008

A two-level program privilege scheme provides security and high reliability. The programs access only their own code and data areas and are restricted from accessing the information which they do not need. Such a scheme not only prevents the deliberate tampering with data but also guards against a faulty program running wild and altering other programs.

The 68008 operates at one of the two privilege levels, the supervisor level or the user level.

At the supervisor level, programs have access to all processor resources and can execute any instruction or access any register. Normally, only the operating system (a collection of specialized programs that control the low level processes within the computer, give application programs a way to communicate with the hardware, and manage the system resources (Pucket, 1985)) runs at this level. All the rest of the software, which includes both the utility and application programs, execute at the user level.

## Data Types And Addressing Modes Used By The 68008

Five basic data types are supported by the 68008. These data types are:

- a) Bits ( Binary digit )
- b) Binary coded decimal digits (4 bits )
- c) Bytes ( 8 bits )
- d) Words ( 16 bits ) and
- e) Long words ( 32 bits ).

There are fourteen addressing modes ( methods by which data or other operand is accessed by the processor ) and 56 instructions ( an operation which the computer can perform in hardware). This is actually less instructions than the older 6800 microprocessor.

The instruction set covers the following classes of operations:

- 1) Data movement
- 2) Integer arithmetic
- 3) Shift and Rotate
- 4) Bit manipulation
- 5) Binary coded decimal
- 6) High level language support
- 7) Program control
- 8) System control and
- 9) Logic.

## Interrupt Structure Used By The 68008

In most applications, programs are seldom executed instruction by instruction without a break. The need frequently arises to respond to an event or exception in the form of an interruption from the hardware or software. Any high performance microprocessor must be able to respond rapidly to a large variety of these interruptions with varying degrees of priority.

The 68008 provides three levels of priority for external hardware interrupts. The supervisor program can put an external interruption ( request for service ) on hold provided the priority level of the interruption is less than or equal to that being serviced.

## Asynchronous Data Transfer Operations In A 68008 Based System

The 68008 is designed to communicate asynchronously on the system bus; that is without a timing signal and with any amount of time between data bytes or words. To indicate when data has been received or sent, the device with which the 68008 is communicating sends an acknowledgment signal to the 68008. This way, the device and the 68008 can operate at different rates and still communicate with each other,

because one waits for the other to finish reading or writing (Carter and Bonds, 1984).

The data transfer process between the processor and memory or peripheral will usually require the following steps:

- 1) Address Decoding,
- 2) DIACK Generation and
- 3) Usage of Interrupts.

#### Address Decoding

The address decode logic generates chip select signal to the appropriate chips ( RAM, ROM and Peripherals ) based on the addresses being accessed. The address decoder used is a four-to-sixteen decoder (74LS154) which is enabled by the 68008 address strobe (AS) signal.

The four most significant address lines A16 to A19 are decoded, splitting the one megabyte memory map into sixteen 64Kbyte (1Kbyte = 1024 bytes) blocks. Some of the devices may require less than 64Kbytes; as a result some of the devices may be addressed repeatedly throughout its 64Kbyte block. This may seem a waste of address space but it allows adequate memory capacity while greatly simplifying the address decoding scheme. The memory map configuration is as shown below:



- RAM is located at addresses 00000 - 9FFFF
- ROM is located at addresses A0000 - AFFFF
- Address locations B0000 - CFFFF are empty
- VDP is located at addresses D0000 - DFFFF
- I/O is located at addresses E0000 - EFFFF
- VPA is located at addresses F0000 - FFFFF

In an asynchronous system, such as the system in this design, the address strobe (AS) allows an output to be enabled only when a valid address appears on the address bus. Valid data is indicated by the data strobe signal being low.

One of the enable inputs to the address decoder is an interrupt acknowledge (IACK). This input is designed to inhibit any other outputs during an interrupt acknowledge cycle. Without this inhibit signal, output Y15 of the address decoder will be selected to cause the reading of the synchronous bus at the same time as the interrupting device is placing its vector on the data bus.

#### Data Transfer Acknowledge (DTACK) Generation

To satisfy the requirements of asynchronous bus transfer, an acknowledgement signal - DTACK, must be sent to the processor by the memory or peripheral to inform it that

the transfer is complete. If necessary, the processor inserts wait states in the cycle until it receives the acknowledgement. Since this design is geared towards educating the new learner, simplified simulations of the DTACK signals are generated using easily understandable logic circuits-the flip-flop.

The three possible sources of acknowledgment signal-DTACK are Random access memory(RAM), Read only memory(ROM) and Peripherals.

Peripheral devices in the 68008 family have outputs which are directly connected to the processor's DTACK input. Memories however do not have such an output so an equivalent signal must be created. Also since cost-effective design was one of the goals of this project, the simulation was designed to accomodate various kinds of memory chips from different manufacturers.

#### Read Only Memory (ROM) DTACK Generator

A quad D-type flip-flop acts as the DTACK generator for the ROM as shown in Figure 3. The ROM chip select signal (ROMOSEL) releases the flip flop from its cleared state which allows a logic zero to propagate from one Q' output (inverted output) to the next on successive rising edge of the system clock (8MHz). The fourth Q' output generates the active low DTACK which signals the 68008 to read the data on

the data bus and to terminate the bus cycle. The DTACK delay time is governed by the number of flip-flops and the frequency of the system clock, and is chosen to suite the ROM access time of the ROM or EPROM being used.

The R/W signal is included in the ROM chip select logic to enable the detection of illegal operations such as write-to-ROM.

#### Random Access Memory (RAM) DTACK Generator

Since many small systems will not require much more than 64Kbytes of system memory which can easily be supplied by Static ram (SRAM), Static and Dynamic ram DTACK generators are discussed. The DTACK logic for the SRAM is different from that of the Dynamic RAM (DRAM) providing an effective design alternative if there should be the need for such decision in the future. In addition, the SRAM is used partly for stack ( a data structure in which the last item added is the first item removed (Williams, 1985) ) operations.

Static ram DTACK generator is similar to that of the ROM except that the R/W signal connects directly to the ram or rams and is not included in the chip select logic as shown in Figure 3.

A Dynamic ram generator on the other hand is much more complex than the two generators described above because it

requires refreshing and address multiplexing logic. The circuit to implement a dynamic ram DIACK generator is as shown in Figure 4. In this circuit diagram, the 18 low order address lines are multiplexed together using three quad two-input multiplexers (74LS157) to form the row and column addresses needed by the DRAMs. Like the ROM and SRAM circuits, a 74LS175 quad D-type flip-flop is used as the DIACK generator. When the ram and rom have the same data access times, a single generator can be shared among these devices. Most of the benefits of an asynchronous bus will be lost if a single generator is shared among several devices having different access times since the processor executes an un-necessarily long bus cycle for the faster devices.

A trade-off can be made between the number of components and the un-necessary bus cycles in a very price conscious design. For Educational purpose, clarity takes precedence hence each generator shows its D-type flip-flop eventhough the ROM and RAM generators could have been generated from a single flip-flop by using the inverted and the non-inverted outputs of the flip-flop.

The quad flip-flop (74LS175) generates the RAM DIACK and also provide the DRAM control signals, row address strobe (RAS), column address strobe (CAS), write (W), and the switching signal (SEL) for the multiplexers. The address strobe (AS), read/write (R/W) signal, and data strobe (DS) are used to ensure the DIACK waveform is generated correctly for the read, write and read-modify-write bus cycle. Figures

5.1, 5.2 and 5.3 show the flow charts for read-modify-write, byte read, and byte write bus cycle operations respectively. Refreshing of the DRAMs occur during the RAS bus cycle. This can be done either in software or with a simple refresh circuit utilizing a dual flip-flop and a timer.

### Usage Of Interrupts

The 68008 has two modes of interrupt processing: User-vectorized interrupt, where the interrupting device provides a vector number on the data bus, and Auto-vectorized interrupt, where a vector address is fetched from memory. The two interrupt pins IPL0 and IPL2 are internally connected together in the 68008 thereby providing only priority levels 2, 5, and 7. Level 7 is a non-maskable interrupt (an interrupt that cannot be inhibited through software).

The interrupt handling logic must assign priority to incoming interrupt requests and route the interrupt acknowledge (IACK) back to the appropriate interrupt source. An eight-to-three priority encoder (74LS148) handles interrupt priority in this design.

A three-to-eight demultiplexer (74LS138) generates IACK signals for the interrupting devices. The demultiplexer is

enabled when address strobe (AS) is asserted and the function codes, FC0 through FC2 are all high (indicating an interrupt acknowledge cycle). The address lines A1, A2 and A3 contain a three bit binary number corresponding to the interrupt priority level being acknowledged.

Interrupt requests arriving at the processor do not force immediate exception processing, but are made pending. Pending interrupts are detected between instruction executions. If the priority of the pending interrupt is lower than or equal to the current processor priority, execution continues with the next instruction and the interrupt exception processing is postponed.

If the priority of the pending interrupt is greater than the current processor priority, the exception processing sequence is started at the end of the current execution cycle.

In the user vectored interrupt mode, the processor responds to the interrupting device by placing the level number of the interrupt on bits A1 through A3 of the address bus and driving the function codes (FC0-FC2) high. The interrupting device then must place a vector number on the data bus and pull DTACK low to signal the 68008 that the vector number is available. The 68008 uses this vector number to acquire the service routine address from the vector table in memory. The content of the interrupt vector whose vector number was previously obtained is fetched and loaded into the program counter, and normal instruction

execution commences in the interrupt handling routine. A flowchart for the interrupt acknowledge sequence is given in Figure 5.4.

In the auto-vector interrupt mode, the processor again places the level number of the interrupt on A1 through A3 and drives the function codes high. Then, instead of placing a vector number on the data bus, the interrupting device pulls the VPA processor pin low. This causes the processor to acquire the service-routine address from the position in the vector table that corresponds to the interrupt level.

This design acknowledges to various kinds of interrupts: interrupts from the keyboard, Serial Input/Output, Printer, Disk controller and other peripherals connected to the I/O decoder shown in Figure 10.

For 68000-type peripherals such as the communication interface (DUART), and the disk controller (IMDC), the level of interrupt acknowledge (IACK) is connected to an input pin on the peripheral for this purpose (eg. IACK on the DUART).

Non-68000 type peripherals such as synchronous peripherals and other processors which are not capable of these vectored interrupt method, need hardware to provide the VPA signal. Asserting VPA during IACK signals the 68008 to use the auto vectors to find one of the appropriate interrupt handlers.

Devices may be chained externally within interrupt priority levels, allowing an unlimited number of peripheral devices to interrupt the processor.

## Synchronous Data Transfer Operations In A 68008 Based System

Two signals on the 68008 processor provide MC6800 type or synchronous interface. They are: enable (E) clock signal and valid peripheral address (UPA) signal.

One output from the address decoder that does not result in the DTACK signal being generated is the Valid Peripheral Address (UPA) bus select signal. This signal requires a synchronous bus transfer.

In addition, a valid memory address (UMA) signal must be provided. Chip select for the synchronous peripherals is derived by decoding the address bus conditioned by the UMA signal. The 68008 does not provide the UMA signal. The UMA signal indicate to the synchronous peripheral that there is a valid address on the address bus and that the processor is synchronized to the E clock. This signal is produced by the circuit in Figure 6.2. The UMA signal in this circuit responds only to a UPA input which signals the processor that the address on the bus is that of a synchronous device and that the bus should conform to the transfer characteristics of a synchronous bus.

To initiate the transfer, the processor input UPA goes low when the decoder UPA signal is low. When the 68008 senses the UPA, it switches into an operating mode in which its overall operation resembles that of an 8-bit 6809



processor. In this mode, the 68008 synchronizes data transfer with I/O devices by asserting the clock signal E. For a write transaction, the rising edge of E indicates that valid data is on the data bus; for read transaction, a falling edge indicates that the processor has latched data on to its informed data bus. At this point, the 68008 also reduces its clock speed to accommodate the slower 8-bit processor and I/O devices. Figure 6.1 is a flowchart of the interface operation between the 68008 processor and the 6800-type synchronous devices. The direction of data transfer is controlled by the R/W signal as shown on Figure 6.2.

#### Synchronous/Asynchronous Data Transfer Operations In A 68008 Based System

Running the 68008 in a synchronous/asynchronous mode allows the synchronous peripherals to be clocked at their highest operating frequencies using an externally generated enable (E) clock. In this mode, the 68008 runs asynchronously and the DTACK signal is generated synchronously with the external E clock. This reduces the number of wait states per bus cycle. This approach requires more circuit components but results in increased system throughput. Figure 7 shows the circuit of the interface logic. In this circuit, the 68008 data bus and the

synchronous data bus are connected via a pair of octal latches (74LS373) joined back-to-back. Their enable outputs are controlled by the 68008's R/W control signal so that one latch is enabled for a read and the other enabled for a write. The latches become latched only when memory/peripheral is deselected. They remain latched until DIACK is negated in the 68008 bus cycle state S7 as shown on Figure 8 (Barth, 1983).

The DIACK signal is generated by the quad JK flip-flop (74LS112) from the data select (DS) signal and the externally generated E clock. The DIACK is then fed to the 68008 processor every bus cycle except during interrupt acknowledge (IACK) cycles. During IACK cycles the interrupting peripherals must issue DIACK in which case the peripheral must provide the interrupt vector number on the data bus, or it must issue UPA signal.

As 6800-type or synchronous peripherals are not capable of generating their own interrupt vector number, DIACK must be suppressed and UPA asserted in its place. The MPSEL signal from the 74LS04 inverter enables the synchronous address decoder logic for the memory and peripheral devices. The relationship between the 68008 and the synchronous bus timing for a read operation is shown in Figure 8. The signal MPSEL is asserted during state S3 corresponding to the beginning of a synchronous cycle. Data from memory or peripheral device becomes valid prior to the falling edge of the external E clock at which time the data

is latched by one of the octal latches, and the bus cycle terminates at the end of S7.

Similarly, during the 68008 processor's write operation, data from the processor becomes valid in S2 and DS in S3. Data is latched by the memory or peripheral device on the next falling edge of the external E clock. The signal DTACK is asserted and the processor proceeds to terminate the bus cycle.

The synchronous binary counter (74LS163) and the jumper block enable the processor and memory/peripherals to operate at several clock rates ranging from 1MHz/2MHz for the memory/peripherals to 4MHz/8MHz for the processor.

## CHAPTER THREE

### COMPARISON OF DATA TRANSFER METHODS AND CONCLUSION

#### ASYNCHRONOUS OPERATION

To achieve clock frequency independence at a system level, the MC68008 can be used in an asynchronous manner. This entails using only the bus handshake lines (AS, DS, DTACK, VPA, BERR, and HALT) to control the data transfer. Using this method, the address strobe (AS) signals the start of a bus cycle and the data strobes are used as a condition for valid data on a write cycle. The memory or peripheral device then responds by placing the requested data on the data bus for a read cycle or latching data on a write cycle and asserting the data transfer acknowledge (DTACK) signal to terminate the bus cycle. If no memory or peripheral responds or the access is invalid, external control logic asserts the bus error (BERR), or BERR and HALT signal to abort or rerun the bus cycle.

The DTACK signal is allowed to be asserted before the

data from a memory or peripheral device is valid on a read cycle. The length of time that DTACK may precede data is given as parameter DL and it must be met in any asynchronous system to insure that valid data is latched into the processor (Motorola, 1985). This parameter is shown on the read cycle timing diagram of Figure 9. On this figure, there is no maximum time specified from the assertion of AS to the assertion of DTACK. This is because the processor will insert wait states or cycles of one period each until DTACK is recognized.

#### Synchronous Operation

The role of the clock in a synchronous system is to connect sequence and time. The interval between clock transitions, whether these transitions are on one wire or distributed over several wires, must be such as to permit enough time for the activities planned for that interval (Mead & Conway, 1979).

To allow for those systems which use the system clock as a signal to generate DTACK and other asynchronous inputs, the asynchronous input setup time is given as parameter SL in Figure 9. If this setup is met on an input, such as DTACK, the processor is guaranteed to recognize that signal on the next falling edge of the system clock. However, if the input signal does not meet the setup time, it is not

guaranteed not to be recognized. In addition, if DTACK is recognized on a falling edge, valid data will be latched into the processor (on a read cycle) on the falling edge provided that the data meets the setup time given as parameter SL1 shown in Figure 9. Given this, parameter DL may be ignored. If DTACK is asserted, with the required setup time, before the falling edge of S4, no waiting states will be incurred and the bus cycle will run at its maximum speed.

## CONCLUSION

Synchronous systems are by far the best known and most widely used. However, synchronous systems possess some serious limitations which are made even worse as chips become larger (Seitz, 1979).

Some of these limitations are physical in nature and relate to the difficulties of moving information from point to point within a single clock period. Another limitation is the difficulty of managing very large designs in a framework in which all system parts must operate together. The same consideration of managing the design of very large integrated systems that provide a motivation for dividing a system into modular parts argue that the parts be independently timed. If the parts are each synchronous

systems with independent clocks, information communicated from one part to another must be synchronized to the receiver's clock. Unfortunately, this synchronization cannot be accomplished with complete reliability. The reason is that synchronizing elements are bistable and have a metastable, or balanced condition that occurs under the conditions in which synchronizers must operate. There is no bound for the time the bistable element may remain in this metastable conditions (Mead & Conway, 1979). This boundless condition makes it fairly unreliable to synchronize the independent parts.

Finally, there is synchronization failure that does occur with synchronous systems. A bistable element in a self-contained synchronous system never has the opportunity to reach a metastable condition, since satisfaction of the timing constraints assures that the output is driven to a voltage outside of the metastable range. But is any system self-contained? A system such as a microprocessor may be entirely synchronous internally but cannot extend this synchronous condition indefinitely to encompass all of the external world with which it may interact. If asynchronous signals of external origin are allowed to enter a synchronous system as ordinary inputs, the timing constraints required to assure correct operation cannot be satisfied, since there is no known relationship between the timing of the asynchronous inputs and the clock (Mead & Conway, 1979).

The limitations imposed by the synchronous discipline suggest that other discipline be tried. One such discipline is the asynchronous or self timed discipline in which the temporal control is delegated to the participating elements of the system.

Asynchronous systems are interconnections of parts which are called elements. Time and sequence are related inside elements, so that events such as signal transitions at the terminals of an element may occur only in certain orders.

Elements can be thought of as performing computational steps whose initiation is caused by signal events at their inputs and whose completion is indicated by signal events at their outputs. The sequencing of the computational steps is determined by the way in which elements are interconnected. The time required to perform a computation is determined by the delays imposed by the elements between initiation and completion, and by interconnection delays or propagation delays. Because of the way asynchronous systems operate, additional circuitry as compared to synchronous systems are normally necessary to check for error conditions of the system bus. These circuits are often called watchdog timers.

A compromise can be reached between synchronous and asynchronous disciplines by combining the two disciplines as discussed earlier. Usually, cost and purpose will determine which discipline is adopted. Since the purpose of this design is educational, all the three disciplines namely:



synchronous, asynchronous, and synchronous/asynchronous are treated fully.

## SELECTED BIBLIOGRAPHY

Barth, Andy

1983 "Using the 68008." Wireless World. Quadrant House  
(November): 70-72.

Carter, Edward and Bonds, A.

1984 "The UU68K Single Board Computer." Byte. Byte  
Publications Inc. (January): 403-416.

Dougherty, Robert

1986 "Memory Mapping Reduces Component Count." EDN.  
Cahners Publications (April): 206.

Mead, C. and Conway, L.

1979 "System Timing." Introduction to VLSI Systems.  
Addison - Wesley Publication Co.: 233-234.

1979 " Properties of Cross Coupled Circuits."  
Introduction to VLSI Systems. Addison -Wesley  
Publication Co.: 26-28.

1979 "System Timing." Introduction to VLSI Systems.  
Addison-Wesley Publication Co.: 236-237.

Motorolla Semiconductors

1985 "Asynchronous versus Synchronous Operation." MC68008  
8-/32- Bit Microprocessor with 8-bit Data Bus.  
Motorolla Inc. (April): 4.28 - 4.30.

Puckett, Dale and Dibble, Peter

1985 "The Historical Connection." The Complete Rainbow  
Guide to OS9. Falsoft Inc.: 1 - 3.

Seitz, Charles

1979 "Self Timed VLSI Systems." Proceedings of the  
Caltech Conference on VLSI (January).

Williams, Steve

1985 "Glossary." Programming The 68008. Sybex Inc.: 473 -  
492.

APPENDIX A

GLOSSARY OF TERMS

## GLOSSARY

AS - Address Strobe. This three-state signal indicates that there is a valid address on the address bus. It indicates the beginning of a memory access. It is also used to lock the bus during the read-modify-write cycle used by the test and set (TAS) instruction.

ADDRESSING MODE - Also known as the Effective Address. On the 68008, this addressing mode is one of several techniques for obtaining data for an instruction. Data may be in a register, in memory, or in the status register. The 68008 has fourteen addressing modes.

BERR - Bus Error. This input informs the processor that there is a problem with the cycle currently being executed.

Problem may be a result of:

1. nonresponding devices,
2. interrupt vector number acquisition failure,
3. illegal access request as determined by a memory management unit, or
4. various other application dependent errors.

CPU - Central Processing Unit. The CPU is a combination of Control Unit (CU) and the Arithmetic Logic Unit (ALU) and is the heart of every computer. The CPU controls a computer's memory and I/O devices. The 68008 is an example of a CPU.

DIP - Dual In Line Package. This describes the packaging

style.

DS - Data Strobe. This three state signal controls the flow of data on the bus. When the R/W line is high, the processor will read from the data bus if DS is low. When the R/W is low, the processor will write to the data bus if DS is low.

DTACK - Data Transfer Acknowledge. This input indicates that the data transfer is complete. When the processor recognizes DTACK during a read cycle, data is latched and the bus cycle is terminated. When DTACK is recognized during a write cycle, the bus cycle is terminated.

INSTRUCTION - An operation which the computer can perform in hardware.

INTERRUPT - An exception caused by an external device.

INTERRUPT MASK - Bits 8-10 of the status register. The 68008 will not recognize an interrupt that is less or equal to the value in the interrupt mask. (NMI may not be masked off in this fashion.)

I/O - Input/Output. The process by which a computer exchanges information with the outside world.

I/O DEVICES - One of several devices, such as a terminal or printer, that can be connected to a computer for purposes of giving information to or receiving from the computer.

K - An abbreviation for "Kilo" (1000), which has been

adapted to mean 1024 in computer terminology.

MULTIIPLEXER - A device that has a number of inputs and a single output and provides a means of selecting the single input which is to be transferred to the output. It can also be used to generate an arbitrary logical function of the selected variables.

NONMASKABLE INTERRUPT - An interrupt that cannot be disabled by software. It is a procedure which forces the CPU to respond to the interrupt regardless of the status of the interrupt disable flag.

OPERATING SYSTEM - A program that controls the execution of other programs and coordinates the function of a computer system.

RAM - Random Access Memory. Also known as Read-Write memory because information can be read from it or written to it.

READ-MODIFY-WRITE OPERATION - A operation that guarantees that a memory flag has been read and modified with no other device allowed to access the flag.

ROM - Read Only Memory. Information can be read from this memory but cannot be written to it.

STACK - A data structure in which the last item added is the first item removed. Also a Last In First Out (LIFO) data structure.

STATUS REGISTER - The 16-bit CPU register in the 68008 containing the trace, supervisor, interrupt mask, and condition code bits.

SUPERVISOR BIT- Bit 13 in the status register, which governs the execution of the privileged instructions.



APPENDIX B

SYSTEM BLOCK DIAGRAM AND PROGRAMMING MODEL

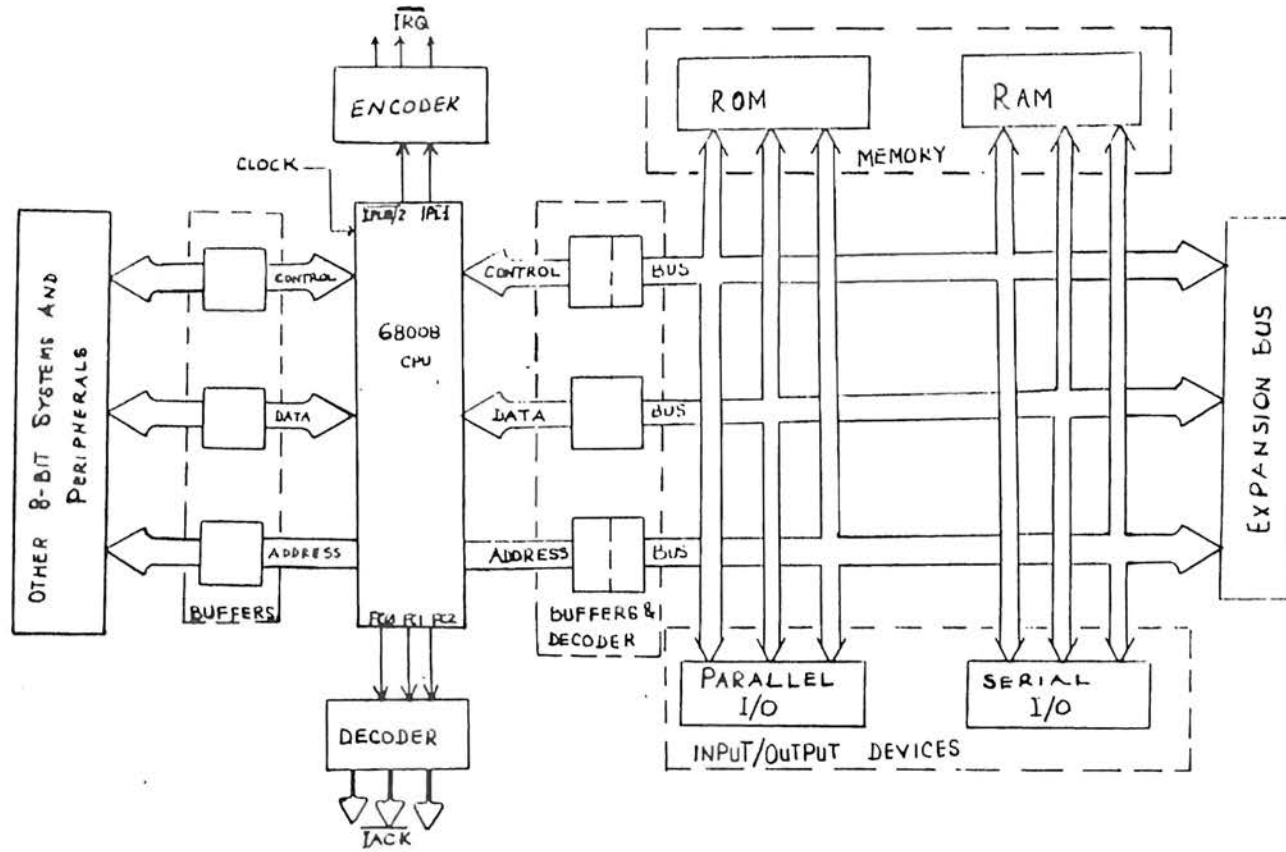


Figure 1 - System Block Diagram

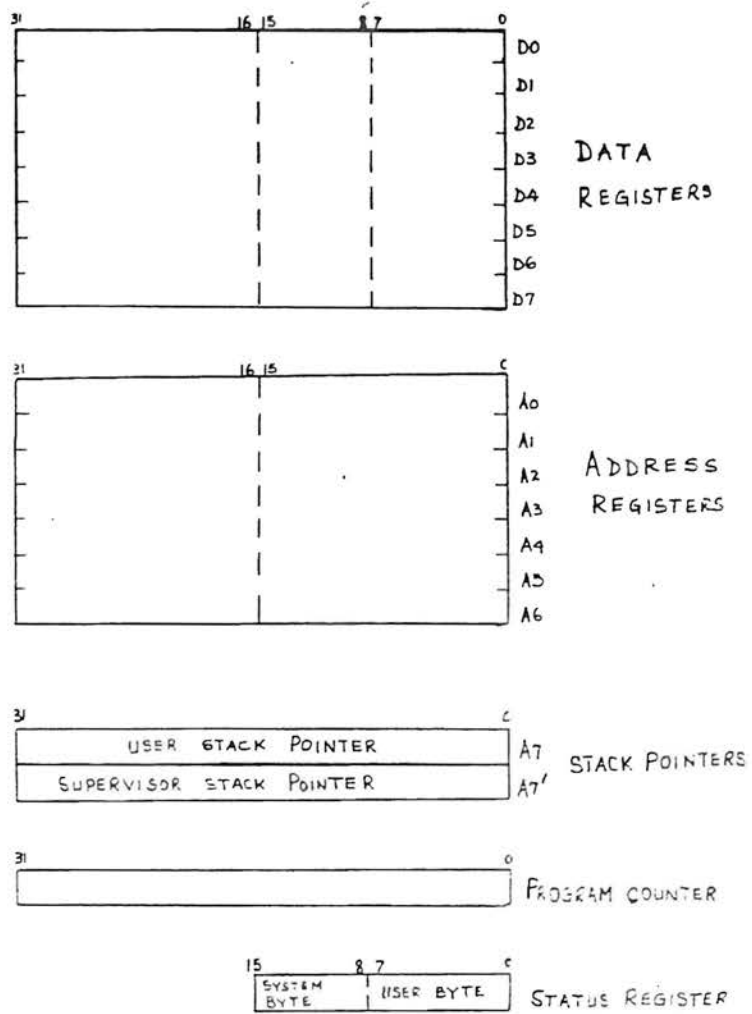


Figure 2 - Processor Programming Model

APPENDIX C

MEMORY DIACK GENERATION

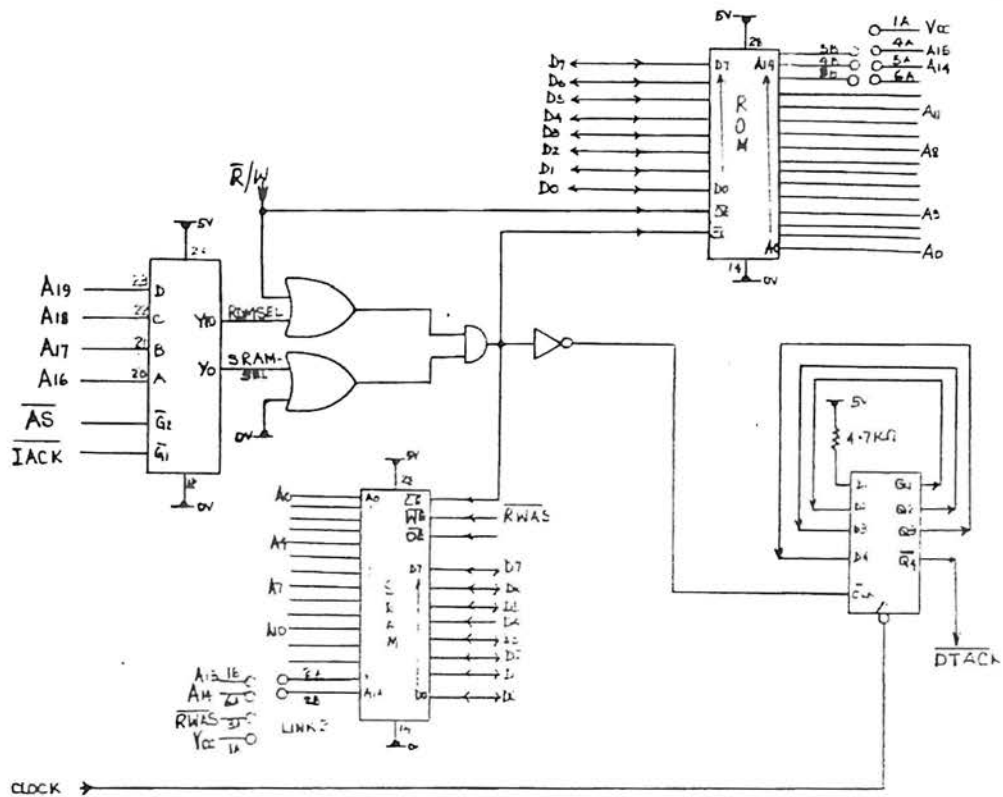


Figure 3 - ROM/SRAM DIACK Generator

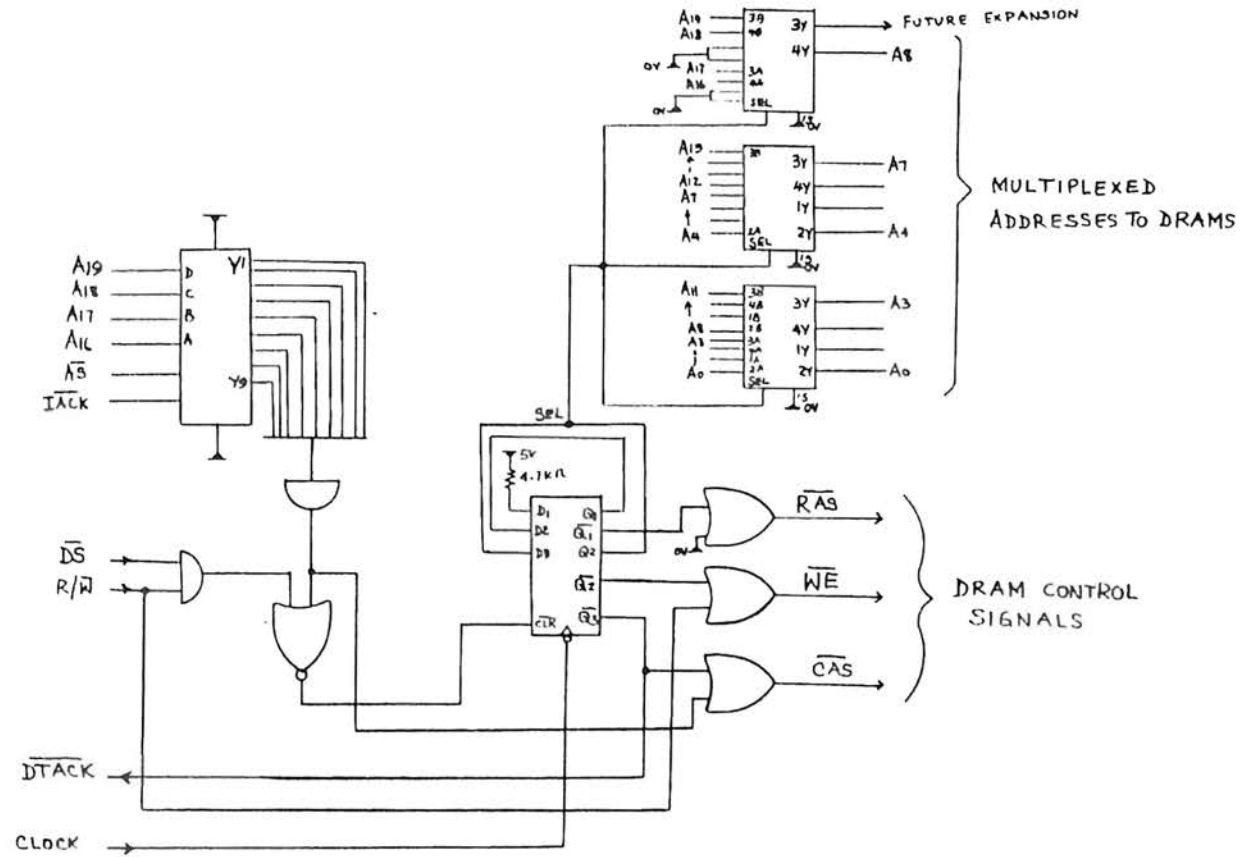


Figure 4 - DRAM DIACK Generator

APPENDIX D

SYSTEM CYCLE FLOWCHARTS

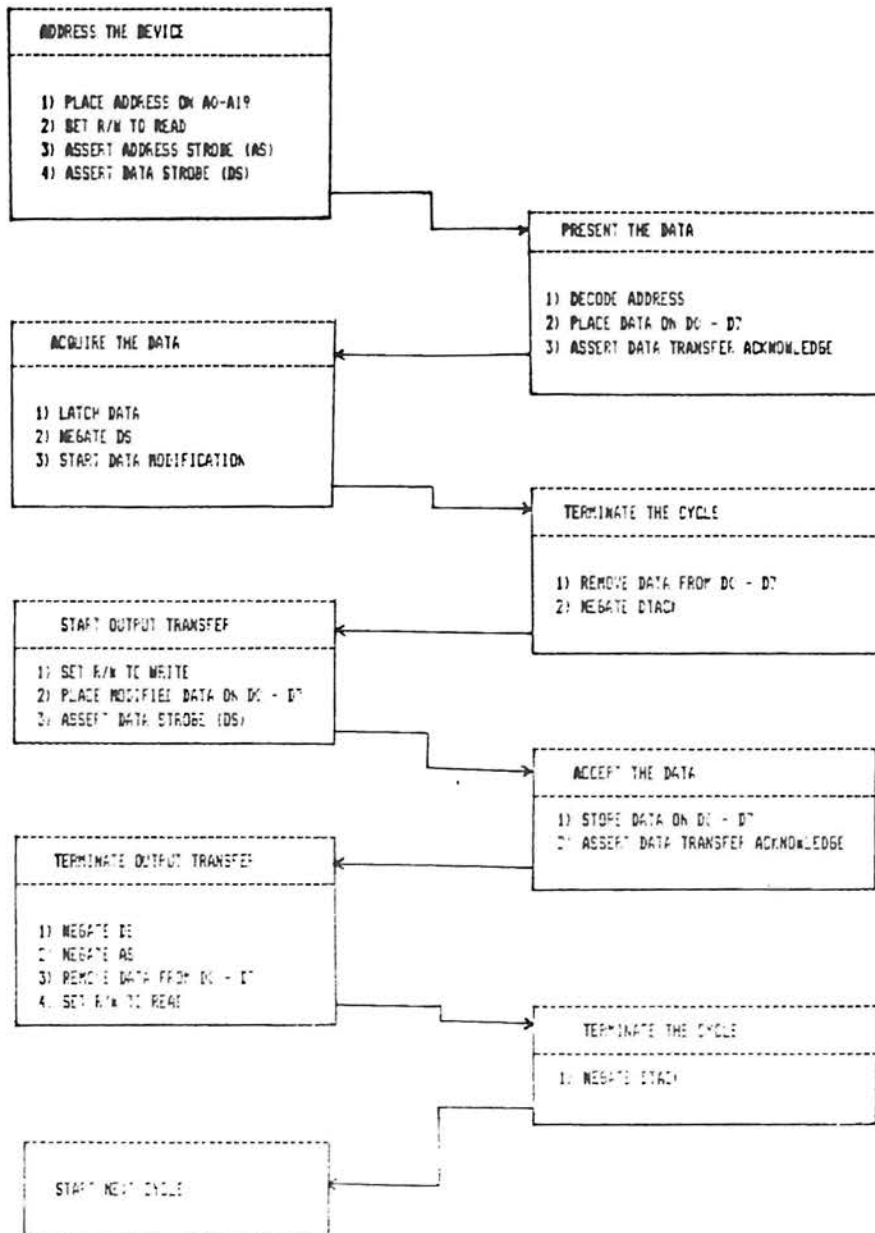


Figure 5.1 - Read-Modify-Write Cycle Flowchart



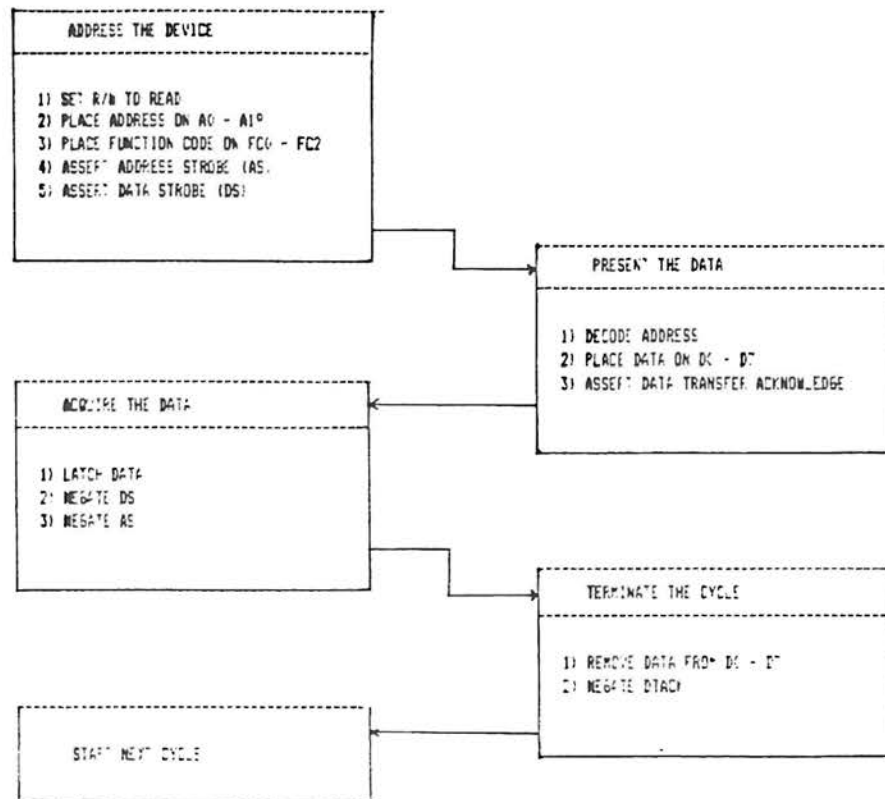


Figure 5.2 - Byte Read Cycle Flowchart

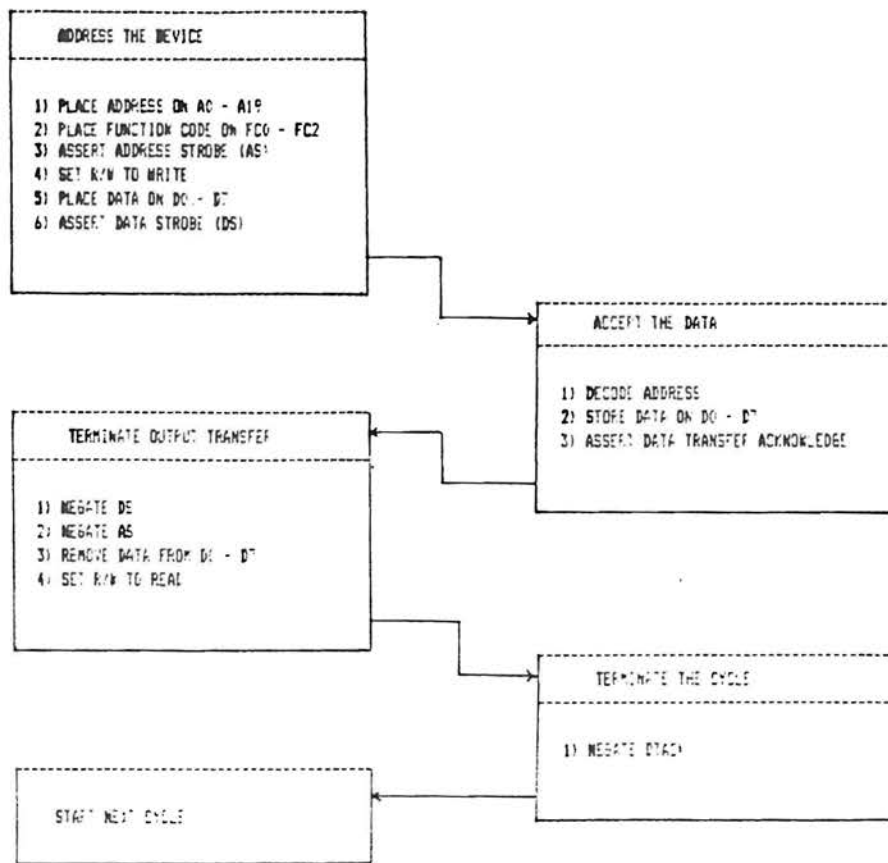


Figure 5.3 - Byte Write Cycle Flowchart

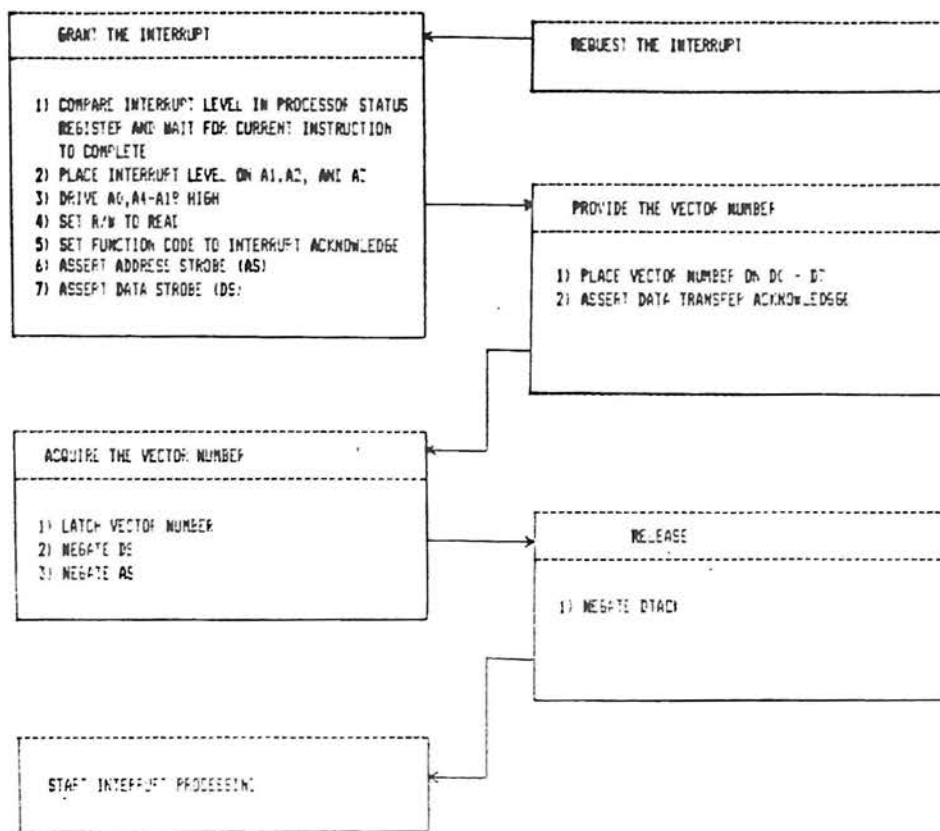


Figure 5.4 - Vector Acquisition Flowchart

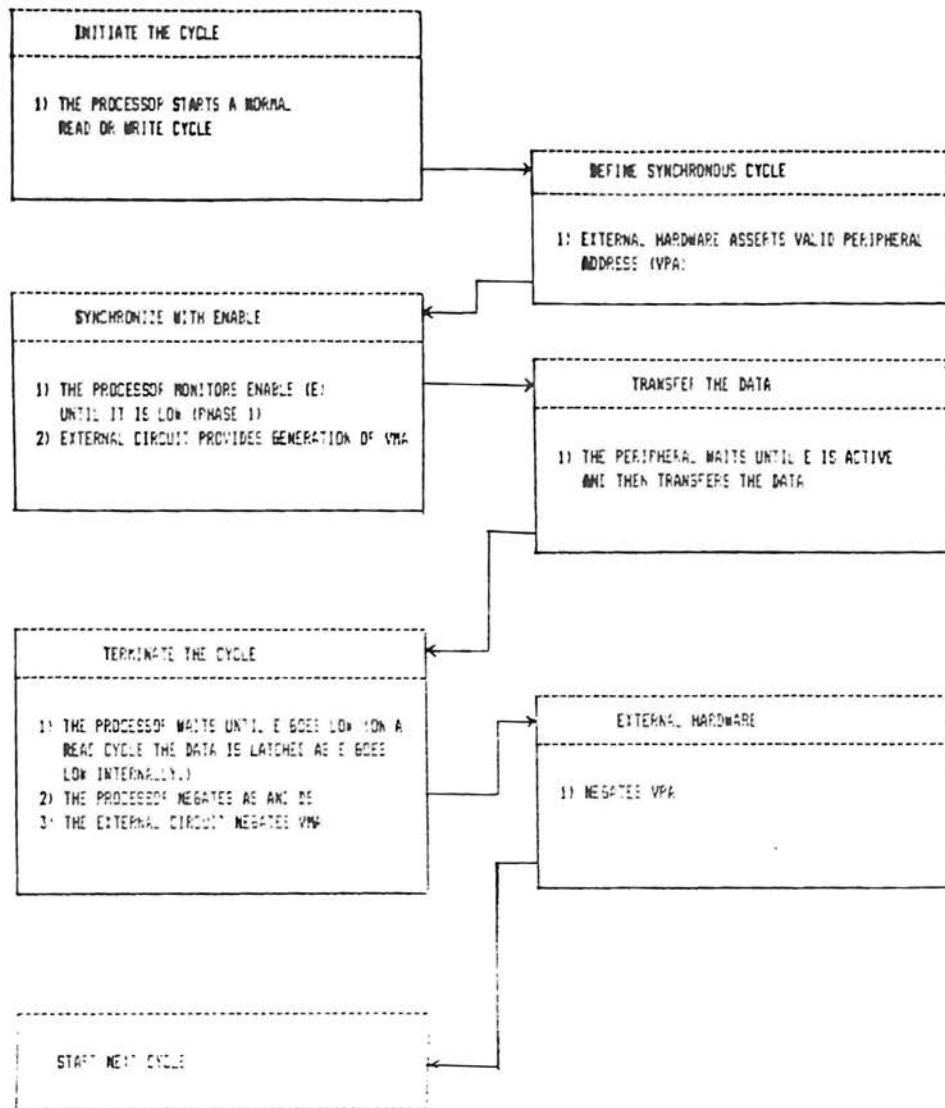


Figure 6.1 - Synchronous Device Cycle Flowchart

APPENDIX E

SYSTEM CIRCUIT AND INTERFACING LOGIC

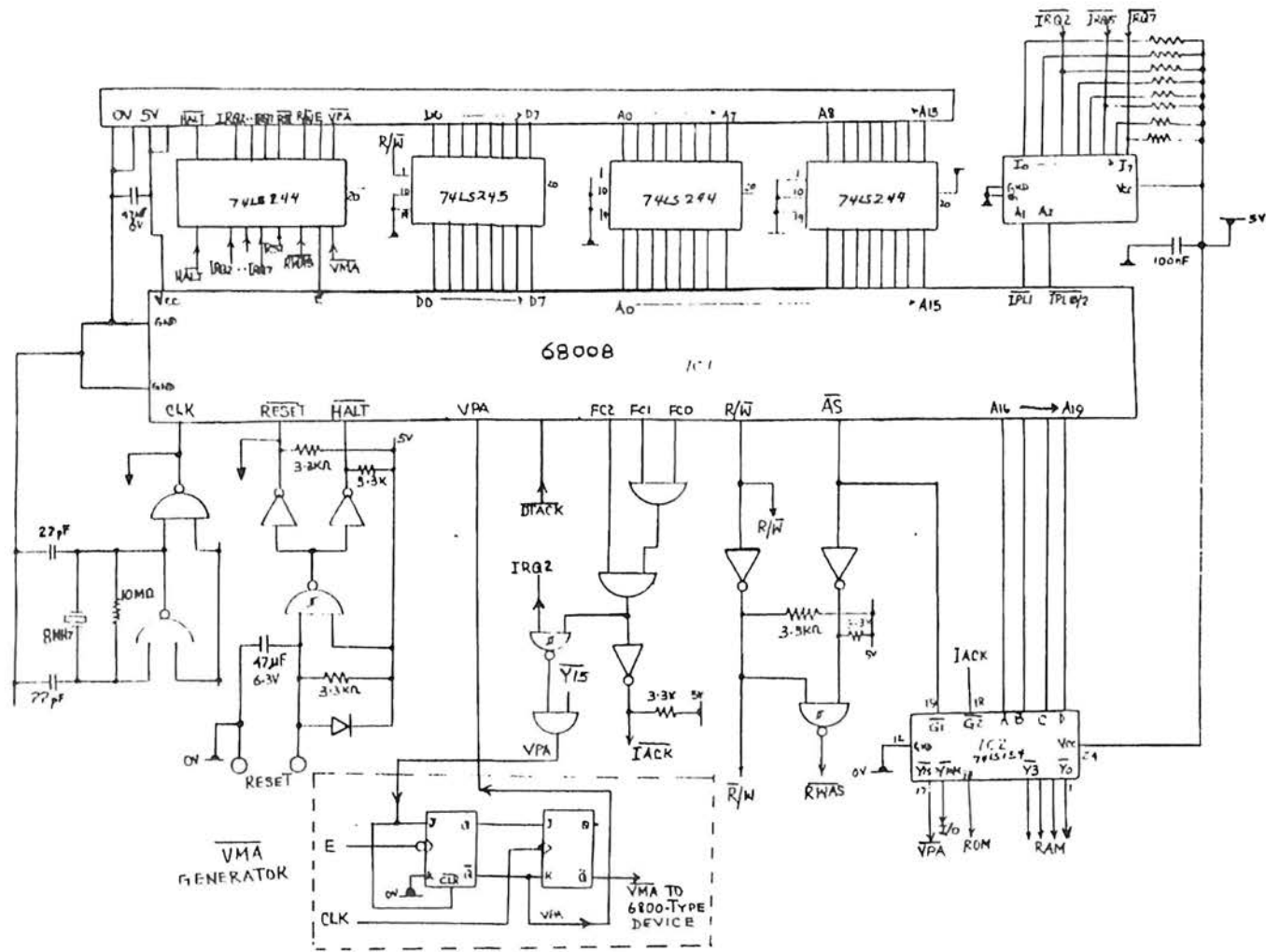


Figure 6.2 - Synchronous Circuit Operation and Interfacing

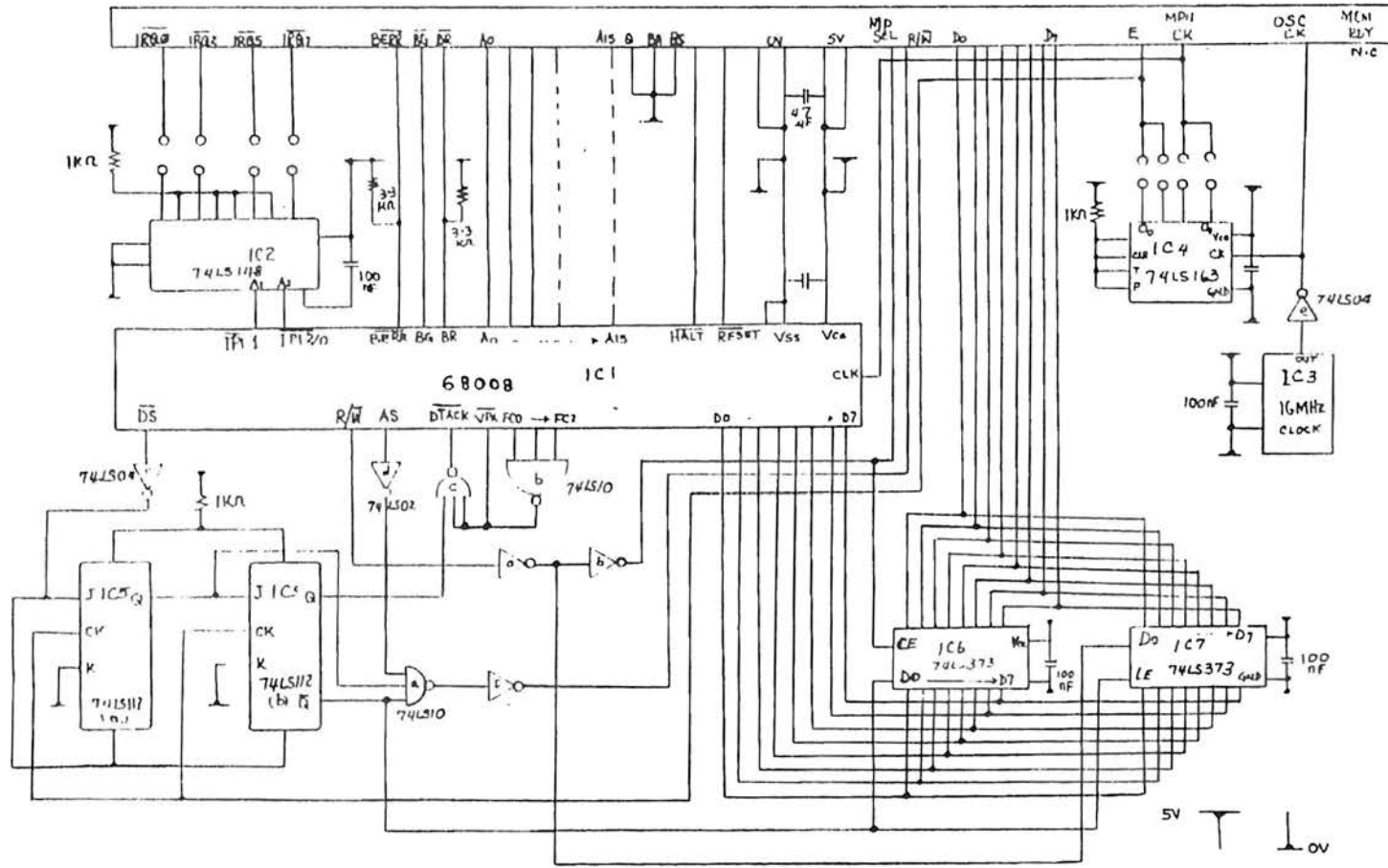


Figure 7 - Synchronous/Asynchronous Interface Logic

APPENDIX F

SYSTEM TIMING AND BUS CYCLE OPERATIONS



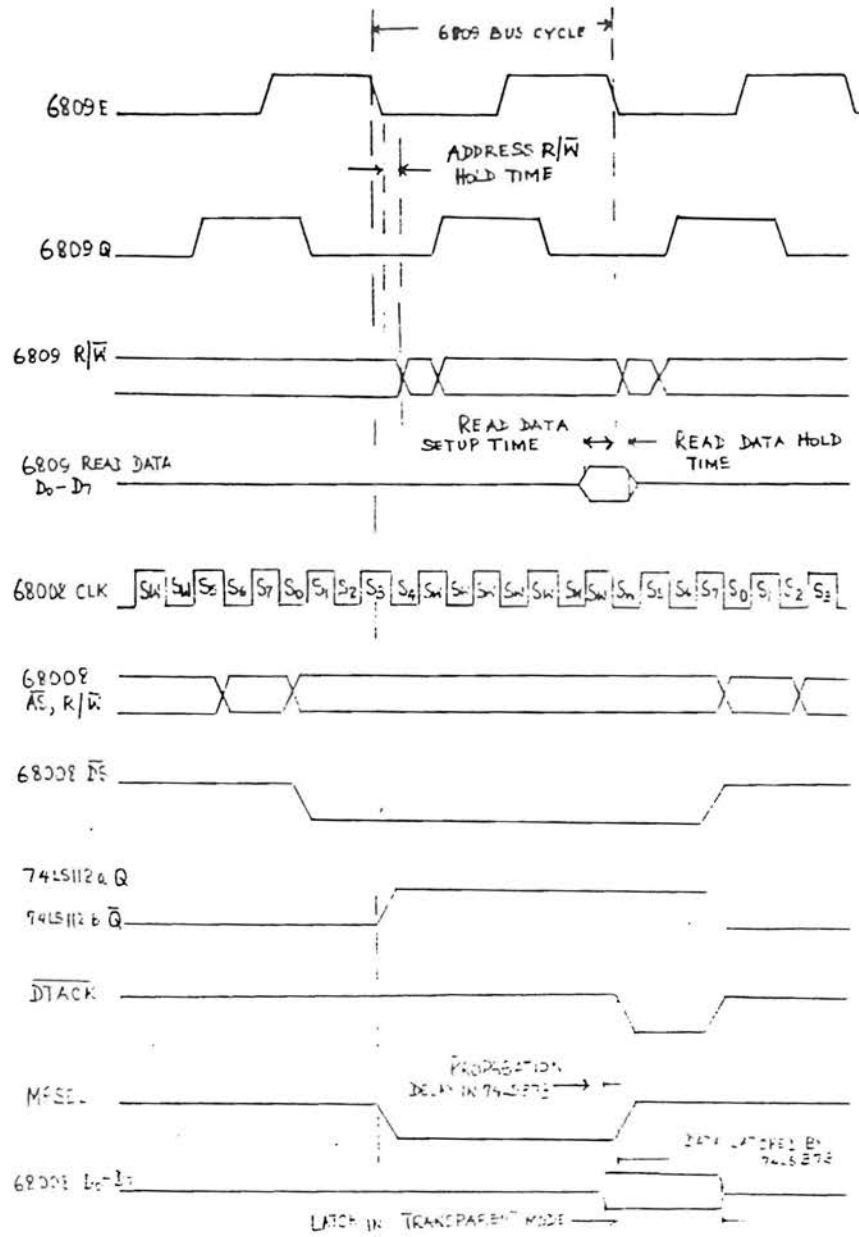


Figure B - MC68008 & 6800-Type Bus Timing for Read Operation

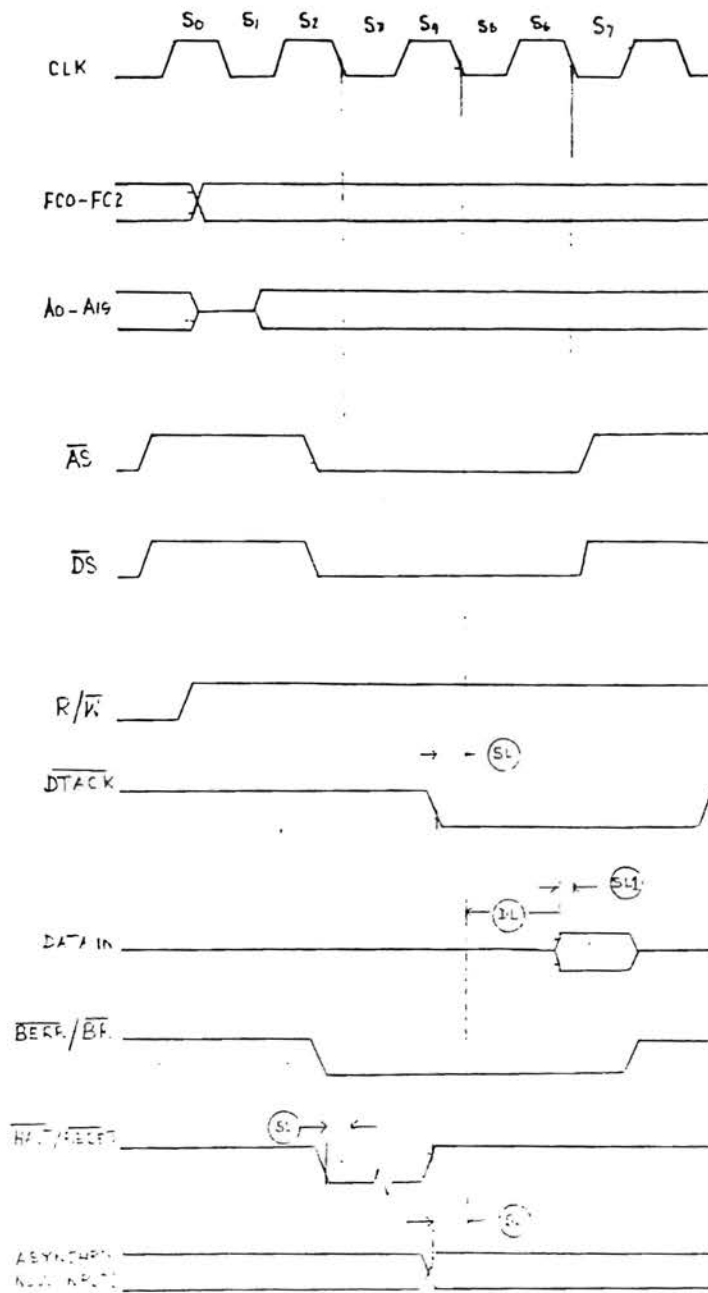


Figure 9 - Read Cycle Timing Diagram for MCE8008

APPENDIX G

INPUT / OUTPUT DECODING

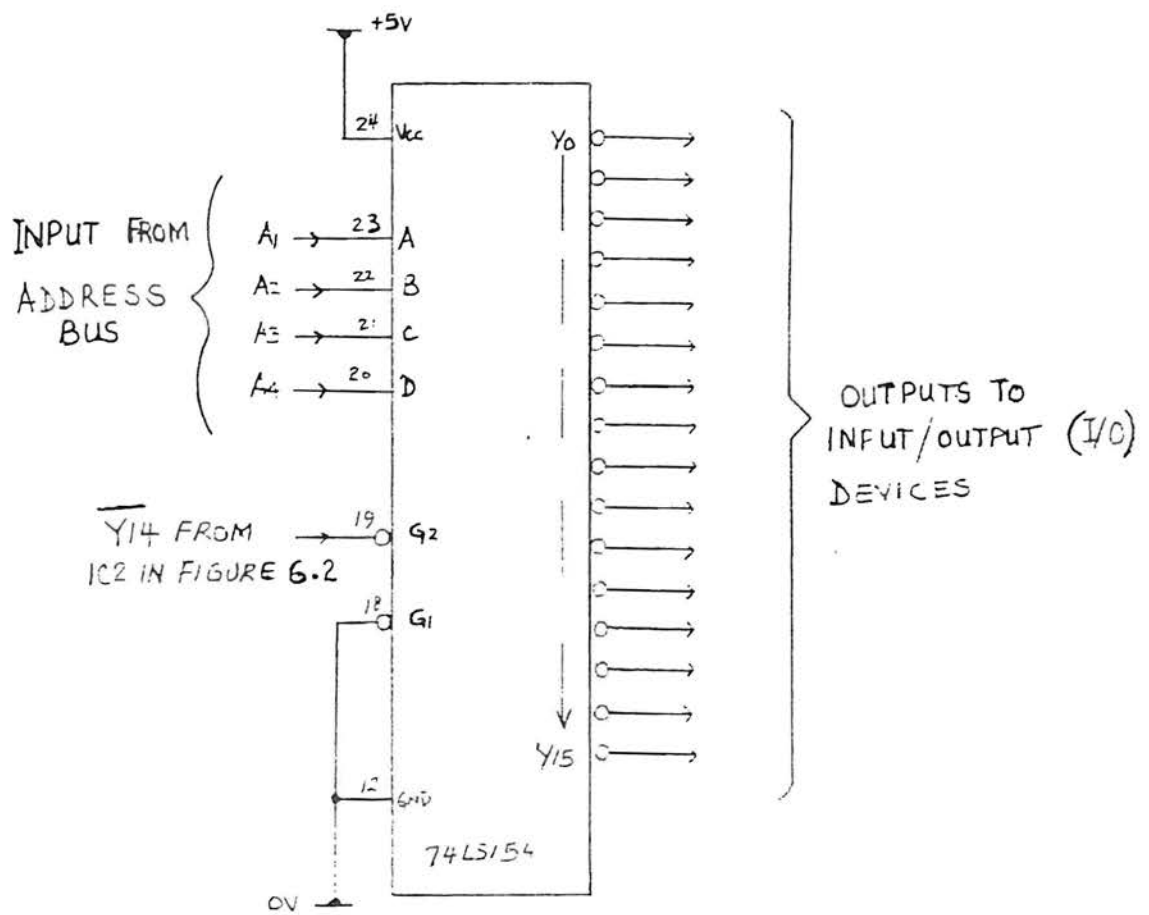


Figure 10 - Input/Output Decoder