1997

A Design in interfacing the MC68HC11 to the AMD AM29F010 flash memory chips

David H. Hands
Edith Cowan University

Edith Cowan University

Copyright Warning

A

DESIGN IN INTERFACING

THE MC68HC11 TO

THE AMD AM29F010

FLASH MEMORY CHIPS

In Co-operation with the Faculty of

Electronic Engineering
EDITH COWAN UNIVERSITY

JANUARY 1997

DAVID HANDS BEng (HONOURS)

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

ABSTRACT

In many environments, motion, vibration and contamination to the Secondary Storage Devices such as hard drives can cause data to become unreadable or even lost. Elimination of these types of magnetic drives, incorporating its replacement with a Solid State Memory Storage Device would provide an invaluable solution for these type of environments.

If a secondary storage system could replace these electro-mechanical disk drive systems incorporating a Solid State Secondary Storage Device such as the Flash Memory Integrated Chips, an increase in the speed of reading from milli-seconds to nano-seconds would transpire as well as providing a robust Secondary Storage Device. In addition to this the rapid increase in the sophistication of software has placed more pressure on the microcontroller to increase its memory capacity, especially that of user RAM. From this need, it is the aim of this thesis to show steps in the designing an interface to the MC68HC11 microcontroller that would increase the user RAM. The design incorporates four Am29F010 Flash Memory Chips as the peripheral Secondary Storage Device.

DECLARATION

I certify that this thesis does not incorporate without acknowledgment any material previous submitted for a degree or diploma in any institution of higher education : and that it does not contain to the best of my knowledge any material previously published or written by another person except where indicated by reference.

ACKNOWLEDGMENT

I would like to acknowledge the supervisory help by Mr. Mike Wetton whom with his aid helped the final prototype of this thesis to fully function.

DAVID HANDS 30/1/1997

TABLE of CONTENTS

PREFACE

This thesis is written with the fore knowledge that the reader has already acquired a basic understanding of microcontrollers, assembly language pertaining to microcontrollers and a background knowledge in computer memory, architecture and digital electronics. The first part of this thesis however, is intended to remind the reader about the fundamentals and basic structures of the MC68HC11 microcontroller and the Am29F010 Flash Memory Chip.

CHAPTER 1

HISTORY Of THE MICROCONTROLLER:

INTRODUCTION

The dawn of the first electronic computer is just over fifty years old. The idea of John Mauchly and J. Presper Eckert was monster-like in look, weighed over 30 tons, occupied $140m^2$, consisted of approximately 18 000 vacuum tubes, 70 000 resistors, 10 000 capacitors and 6000 switches. The machine used 150kW of power and was strangulated by hundreds of miles of wiring (McCrindle, 1990). At that time and only in the minds of science fiction writers would the size of the computer be conceivable as being reduced to a square centimeter and as portable as a dollar coin.

Fifty years later on this fiction became a reality indebted to the advent of the microcontroller and ingenious manufacturing techniques. In the late 1960's, a Japanese company, Busicon, asked Intel to produce 12 chips for a new range of calculators (McCrindle, 1990). The task of designing the chip was placed in the hands of Marcian Hoff, who at the time was working at the Digital Co-operation. His ideas, based on his experience with the PDP-8 computer and new chip technology, led to the development of the first microprocessor. Although the task of producing this chip was left with Hoff, a patent application for a computer on a chip was submitted by Gilbert Hyatt in 1970. In the following year Hyatt and his partners went in separate directions amidst long legal battles. His partners formed a new company and named it Intel. During that year, 1971, the first microprocessor was produced by Intel (Spasov, 1992). The chip called the Intel 4004 contained 2250 transistors and handled binary data as 4-bit words.

By 1974, several chip manufacturers offered 8-bit microprocessors and products controlled by 8-bit microprocessors. The most common of these microprocessors

were the 8080, 8085, Zilog Z8000 and the Motorola 6800. In 1978, 16-bit microprocessors had become common including the Intel 8086 and the Motorola 68000. Since then microprocessor manufacturers have continually been developing microprocessors with advanced features and new architectures.

One of the further developments of the microprocessors was the microcontroller. Microcontrollers were designed to provide all computing functions on a single chip. Although the general public is more aware of the microprocessor due to its popularity in personal computers, more microcontrollers are sold than microprocessors because they are used in many machines, instruments and consumer products (Spasov, 1992).

Playing an integral part in televisions, compact disc players, washing machines, telephones, automated manufacturing systems and the ever complex robotic systems, the microcontroller, despite its power and versatility is a very understated tool in the electronic industry. A leading electronic magazine publishing company has reported that the average home in North American has 35 microcontrollers in domestic appliances and by the year 2000 it is expected that this number will grow to 240 (DataQuest,1994). As a consequence of their every day use microcontrollers have now become an integral part of our everyday lives.

To illustrate the strength in economic terms of the microprocessor and the microcontroller, a company known as Motorola, in 1993, had a 55 percent of share of the microprocessor market that was estimated to be worth 600 million dollars. It also acquired a 10.4 percent share of the microcontroller market worth 484 million dollars (Electronic News, 1994). It is also predicted that by the year 2000 an estimate total of 2,700 million 8-bit and 1,025 million 16-bit microcontroller units will be shipped world wide (WSTS ICE -1994). From 1992 to 1993 market share estimates indicated a 38 percent annual growth for the Motorola microcontroller sale, compared with a 26 percent growth for the industry (EDGE: 1994).

It is now indubitable that microcontrollers are the intravenously linked life line to many electronic applications and in terms of business prospects is worth millions of dollars. Due to the position of the microcontroller in the economic market, any advancement to its architecture would certainly be of high importance and of extreme benefit to a great many applications.

RATIONALE

An extremely valuable development that could be made to the microcontroller is of increasing its limited amount of memory. Due to the rapid increase in the sophistication of software more pressure is placed on the microcontroller to increase its memory capacity, especially that of user RAM. By increasing RAM in the microcontroller architecture, easier handling of these software programs would be ensued. From this need, it is now the aim of this project to design an interface to the microcontroller that would increase the user RAM. The design will incorporate the Am29F010 Flash Memory Chip, details of which will be discussed later.

An associated long term aim of this project is to provide a Solid State Secondary Storage device that will replace the awkward electro-mechanical disk drive system. Due to their mechanical components, magnetic disk drives (including hard drives) are much less reliable than solid state disks. In many environments, motion, vibration and contamination to the hard drive can cause data to become unreadable or even lost. Elimination of these magnetic drives, incorporating its replacement with a Flash Memory Storage Device, would provide an excellent solution for these type of environments.

If a secondary storage system could replace the electro-mechanical disk drive system incorporating Flash Memory, an increase in the speed of reading from milli-seconds to nano-seconds would transpire. Also, the factor of being more robust than the electro-mechanical device offers inspiration to persist with this

aim. Example of applications that would benefit from using this new secondary storage device is listed in Chapter 3 ranging from Digital cameras to complex Aircraft Navigation Systems. The list is ever increasing and makes this project of extreme value.

NOTE : The microcontroller that will be used in this project is the MC68HC11. Edith Cowan University has several HC11 development boards available and staff with expertise in HC11 technology.

In 1985, Motorola developed the MC68HC11 microcontroller which due to its special advancements from the Motorola 6800 and its associated low expense also became the choice tool for this project. As the 68HC11 plays a fundamental role in this project a brief discussion on the microcontroller and its features will be given.

CHAPTER 2

OVERVIEW OF THE MC68HC11

Developed by Motorola in 1985, the MC68HC11 is an 8-bit data bus and a 16-bit address bus microcontroller (Spasov,1992). The HCMOS (high density complementary metal-oxide semiconductor) technology used on the MC68HC11 combines smaller size and higher speeds with low power and high noise immunity of the CMOS (complementary metal-oxide semiconductor). It has a nominal clock speed of 2 MHz with an on-chip memory systems including 8K bytes of ROM , 512 bytes of EEPROM and 256 bytes of RAM.

Major peripheral functions of the MC68HC11 include an eight channel analog to digital converter (ADC). A asynchronous serial communications interface (SCI) and a separate synchronous serial peripheral interface (SPI). There are three input capture lines to the main 16-bit address, a free running timer system, five output compare lines , and real time interrupt functions.

To protect against system errors self-monitoring system circuitry is included on-chip. A COP computer operating watchdog is also included on-chip to protect against software failures. Another feature of the MC68HC11 is a system reset in case the clock is lost or runs to slow. An illegal opcode detection circuit provides a non maskable interrupt if an illegal opcode is detected.

A further aspect of the microcontroller is the register set. In the MC68HC11 there are two types of registers, the CPU registers and the I/O registers.

I/O REGISTERS

There are three basic types of I/O registers; control status and data. Each I/O register holds I/O data associated with its corresponding I/O port. The I/O control register control and monitor the I/O processes of the microcontroller.

Most of the I/O ports are multifunctional. For example Port E of the MC68HC11 can be used as a dedicated 8-bit input port or a channel for a Analog-to-Digital operation. Because of this versatility the option in respect to interfacing more than one Flash Memory Chip is available. The usage of these port will depend on how the associated control and status registers are configured. The ability to control and monitor the I/O ports using these registers is one of the distinguishing characteristic between microcontrollers and microprocessors.

CPU REGISTERS

The CPU registers of the MC68HC11 contains two 8-bit Accumulators (ACCA and ACCB) and a 16-bit Accumulator (ACCD). Two index registers (IX and IY), a stack pointer, a program counter and condition code registers. These registers are used for mathematical computations, storing values to the I/O ports (or to memory) and to monitor the state of the microcontroller.

There are several other features of the MC68HC11 but the purpose of this overview was just to cover the main characteristics of the MC68HC11.

Figure 1 overleaf illustrates the major subsystems of the MC68HC11 and how they relate to the pins of the MCU.

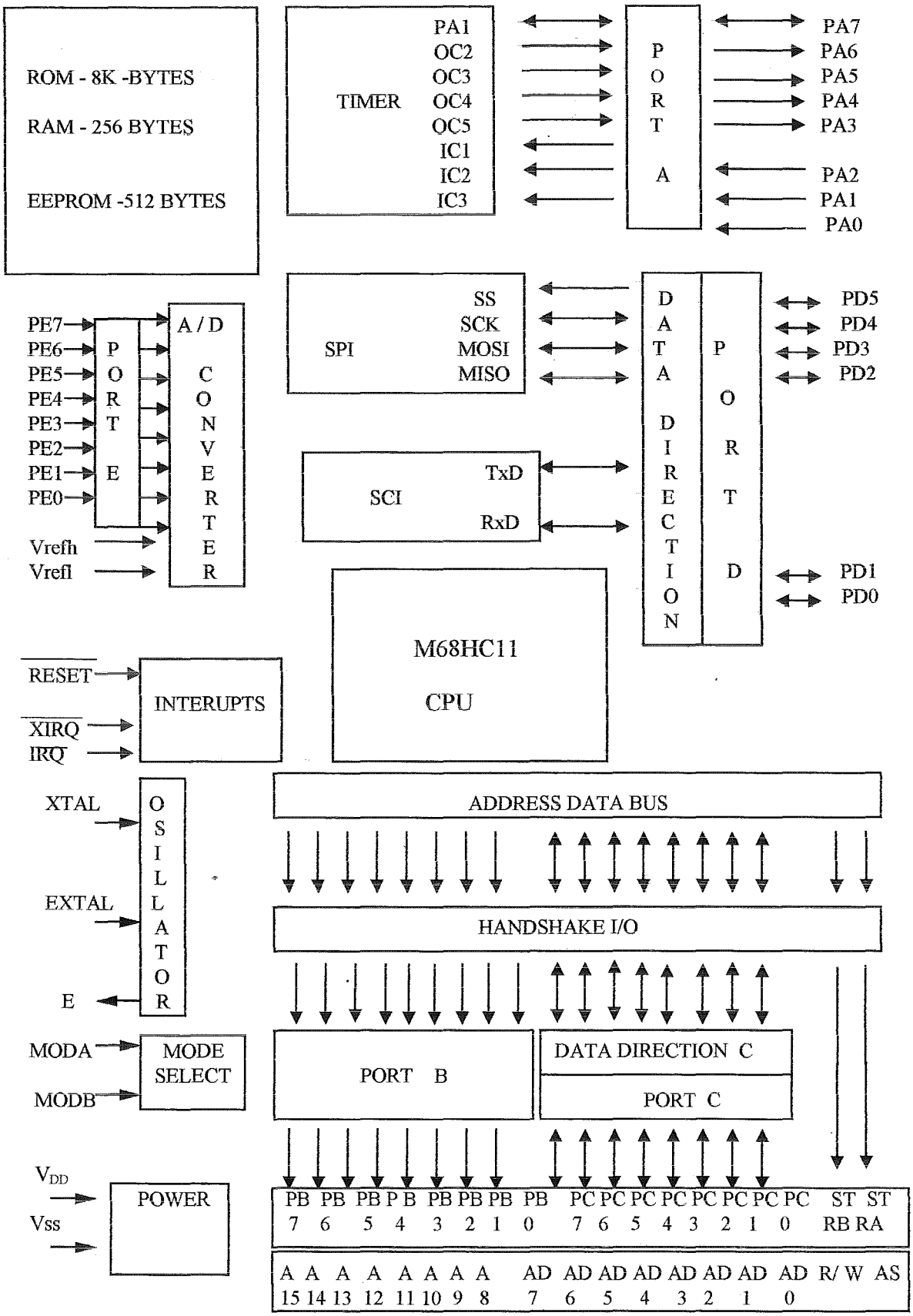


FIG 1 : MC68HC11 OUTLAY THE 68HC11 MICROCONTROLLER

Having established a brief overview of the MC68HC11, a question of why Flash Memory over magnetic memory structures is used to extend the memory of the microcontroller would not be without merit. A discussion answering this question will now be given.

CHAPTER 3

FLASH MEMORY

INTRODUCTION

The MC68HC11 was the first microcontroller to have an internal EEPROM which proved to be a great asset to programmers. Although this was an impressive technological advancement in microcontroller architecture, an obstacle still arose in the way that microcontrollers come with a limited amount of memory. The result of this limited memory now provides a challenge to look for a way to increase the size of this memory. The fundamental task now at hand is to create and design a way to expand this limited memory and interface this extended memory to the MC68HC11. A further challenge that also needs to be under taken is interfacing the MC68HC11 with the new expanded memory to the outside world without stepping over the boundaries of the already well established IDE and SCSI standards.

One type of non-volatile memory expansion system is the Flash-Memory system. Flash memory is the first significantly new solid state memory technology to appear in over ten years (IEEE, 1993). It is nonvolatile, easily update-able and is high in density. Flash Memory has the ability to store information even when power is disconnected. It requires power only when being read from or written to, thus, reducing power loading on portable devices. Flash Memory also offers system designers a wealth of applications and it seems to have no limits. Several applications of Flash Memory are listed below

FLASH MEMORY APPLICATIONS

The most common forms of Flash Memory media storage include:

- DATA : Computer Files, Email, Data Acquisition (2 Mbytes allows you to store 1,200 to 1,400 pages of double-spaced text).

- AUDIO: Voice Recordings, Voice Messages, Sound Clips (2 Mbytes allows storage of 30 to 60 minutes of voice for audio devices).

- IMAGES: Digital Film Video Clips, Scanned Images, Drawing, Writing, Fax (2 Mbytes allows you to store 20 to 24 images for digital cameras).

- COMMUNICATIONS: Voice/data pagers, Cellular phones/accessories, Mobile answering systems, Wireless fax/modems,

- PORTABLE COMPUTING: Flash PC Cards (solid-state Disks) for notebooks and PDAs, Combo and Multi-function PC cards, Portable scanners

- INDUSTRIAL: Talking sales displays, Hand-held terminals/meters, Portable bar-code readers, Data recorders,

- CONSUMER: Digital voice recorders, Digital cameras, Video capture and Pen-based tablets..

Whether talking about removable flash for gathering, transporting and sharing data in a mobile environment or embedded flash for ruggedizing portable PCs, applications for flash technology are expanding. As their is a vast number of

applications that can use the benefits of this technology, the rationale behind this project makes it even more justifiable.

FLASH MEMORY HISTORY and CAPACITY

Although only introduced five years ago, flash memory is already reached its second generation. Flash Memory chips have been produced in volume for the last five years, but are now moving from novelty status into the mainstream (IEEE, 1993).

In 1992, Intel, the world's largest chip maker and Sharp, an internationally recognized leader in industrial electronics products, established their technology partnership to jointly develop and manufacture future generations of flash memory products (EDGE, 1996).

This agreement called for the companies to combine their respective areas of technology, design and manufacturing expertise to foster Flash Memory growth in the computing, communications and consumer electronic markets (EDGE: 1996)

Flash Memory chips are now available up to 64-Mbits but this project is based on standard 16 M-bits in capacities with random access times of 60 ns (Arnold 1996). Combined with complete non-volatility, this capacity makes the chips a potential fit in applications that might otherwise have used ROM, EEPROM, battery-backed RAM, or magnetic mass storage. Flash memory is also in-system update-able: it can be erased a block at a time and is programmable a byte at a time.

FLASH VERSUS OTHER MEMORY STRUCTURES

The redundancy of traditional memory systems both primary and secondary have done little to optimize the performance of the fast microprocessors being produced today. An example of this is the system boot time and the application

task-switching response, which are both severely hampered due to disk drive spin-up time and nonvolatile memory to RAM file-load delays. Whereas Flash Memory has a 60nSec access time.

Embedded-system designs today are revolutionary, likewise, so are their memory architectures. Incremental improvements to the traditional approach no longer measure up to the potential of the flash-memory-based alternative. Flash memory provides both the high read performance of DRAM and SRAM and the nonvolatility of ROM and hard-drive-like write performance. Lengthy software-load overhead and task-switching delays are eliminated. Codes run as fast or faster than that in DRAM with less system-hardware complexity.

In relation to magnetic memory's, Flash Memory has a wider operating-temperature ranges than most magnetic hard drives. These magnet drives also have several moving parts which have a unacceptably low tolerance to vibration, shock, and movement during read/write in rugged or mobile environments.

The price of a high-density Flash Memory array is comparable to that of a DRAM/ROM or DRAM/small hard drive combination. Unlike ROM, Flash Memory is in-system updatable thus, keeping system costs low both initially and throughout system lifetime.

Being nonvolatile, Flash Memory does not have to be constantly refreshed by the memory controller in order to preserve its stored data and thus no need for continuous application of power to retain stored information. When compared to hard drives, Flash Memory consumes less than 5 percent of the power of a disk drive. It is more beneficial in consumer products as most consumer products run on two AA batteries, getting the disk drive up to spin would be near impossible, but Flash Memory would make it possible. Flash Memory also offers a much higher performance, up to 30 percent higher than a typical disk drive in a typical

read-write application: (Balch, K. 1996). Its smaller form factor than disk drives also enable it to go into applications where disk drives are just too big e.g.such as pagers, audio recorders, cell phones, smart cards or medical devices.

Consequently, flash memory is a very efficient memory technology, consuming little system power.

FLASH MEMORY ARCHITECTURE

Flash memories are descendants of EPROM and E2PROM technology, and these building blocks provide still further differentiation of flash products, notably in the way they write and erase data. Flash Meomory is solid state, rewritable and a nonvolatile memory. It does not require a power source to maintain data and unlike ROM it is fast. Although Flash Memory is not new, some of the uses for it are.

A major point of rivalry among the flash proponents is architectures, of which there are now four major forms: NOR, NAND and two new varieties, AND and DINOR (divided bit-line NOR). Each has its advantages and disadvantages.

The EPROM NOR device uses a combination of channel hot-electron injection to write, and Fowler Nordheim tunneling to erase. Channel hot-electron injection is a relatively slow mechanism with high average current, whereas Fowler Nordheim tunneling is fast and uses very low current. E2PROM-based flash devices use Fowler Nordheim for both write and erase, and therefore tend to be much faster than EPROM types and consume less power.

Similarly, the four flash architectural types provide different capabilities. The NOR architecture's parallel structure and fast random-access capability suit it for both program and data storage. The serial structure of NAND architecture, offered

by companies such as Samsung, Toshiba and National Semiconductor Corp., has a relatively slow random access due to a built-in latency between page accesses. But it performs well in data-storage applications such as PCMCIA memory cards and ATA cards.

All of these architecture share the primary characteristics of nonvolatility and updateability, but differ in such secondary characteristics as block size (the number of cells that are erased at one time), access time, and density.

The NOR architecture is simplest, but it requires a dual power supply and its large block size creates challenges for hard-drive emulation. The EEPROM flash device has a small block size of 64 bytes and a single external power supply. The NAND EEPROM flash architecture has an intermediate block size of 4K bytes and includes error detection and correction (EDAC) circuitry.

FLASH SIMPLICITY

A flash memory cell contains only one transistor. This simplified architecture results in significant density advantages over both EEPROM and Static Random-Access Memory (SRAM). Flash memory can be as dense as ROM and emulate dynamic RAM (DRAM).

Today almost every top semiconductor manufacturer of EPROM's, ROM's and DRAM's are today further developing flash memory technology. There are two trends that explain why.

1. Single-transistor flash memory will eventually cost less to make than DRAM because of its simpler cell architecture (flash needs no trench capacitor).
2. There are no fundamental technical limitations that can keep flash from replacing ROM and RAM used for execution-code and data storage.

Note: The detail of these archictetures is not elaborated as it is as extends beyond the purpose of this report..

FLASH MEMORY HOW IT WORKS

When a read occurs in flash memory, address inputs select specific transistors within the multi-kilobit device array. In Intel Corp.'s ETOX technology (which employs the NOR architecture), the decoded address generates supply-voltage levels on the CMOS field-effect transistor select gate and drain, while the source is grounded.

In an erased cell, the select-gate voltage is sufficient to overcome the transistor turn-on threshold voltage (V_T), and the drain-to-source current is detected by the sense-amplifier circuitry and translated into a 1. In a programmed cell, however, added electrons stored on the floating gate raise the transistor's V_T so that applied voltage on the select gate is not sufficient to turn it on. The absence of current results in a 0 at the corresponding flash memory output.

To program a cell, the Intel ETOX flash memory technology, like EPROM, applies 12V between source and select gate (capacitively coupled to the floating gate) and approximately 5V between source and drain. The source-drain voltage generates hot electrons that are swept across the channel from source to drain. Colliding with atoms along the way, these hot electrons create even more free electrons. The high voltage on the select gate overcomes the oxide energy barrier, and attracts the electrons across the thin lower oxide, where they accumulate on the floating gate. When enough have accumulated, the cell switches from its 1 (erased) to its 0 (programmed) state. This programming technique, called channel hot-electron injection, is also used for EPROM cells.

Other flash memory approaches program by using tunnelling from the abstrate to the floating gate and higher internal voltages of up to 25 V. The tradeoff for the faster cell program speed that results from the higher voltage is possible cell unreliability due to oxide breakdown.

To erase a flash memory cell, electrons are removed from the floating gate, returning the transistor to its normal V_T threshold behavior. Subsequent reads from the cell again output a 1. First-generation flash memories were erased in bulk like EPROMs; an erase operation removed the charge from all transistors in the memory array at the same time. Now flash memories erase in smaller blocks, making them more suitable for both code-and file-storage applications.

Intel's ETOX flash memory technology erases by using Fowler-Nordheim tunnelling. Floating the drain, grounding the select gate, and applying 12V to the source creates an electric field across the thin oxide between the floating gate and the source and pulls electrons off the floating gate.

Preconditioning or preprogramming, a block of 0s before erasing puts an equal amount of electron charge on the floating gate of each cell in the block before all are erased in parallel. This technique ensures uniform and dependable erasure, resulting in equivalent threshold voltages for all transistors in the erased array or block, and is essential to reliable device operation as the memory is cycled. Many second-generation NOR flash memories automatically precondition before erasure.

Other flash memory approaches reverse-bias the same substrate/floating-gate junction used for programming, or use a separate floating-gate/erase-gate junction and apply much higher voltages to speed erase time (resulting in higher electric fields). The consequence of this tradeoff is potential oxide breakdown, especially if the cell is repeatedly cycled.

CHAPTER 4

FLASH MEMORY INTERFACE DESIGN.

Semiconductor memory is where data and program code are normally stored in a microcontroller system. Physically, there are different types of memory, such as *ROM, RAM,* and *EEPROM.* The MC68HC11 has an on-chip memory system that includes 8K bytes of ROM , 512 bytes of EEPROM and 256 bytes of RAM. A brief Memory Map is illustrated below in Figure 2. The memory map shows the outlay of the various memories and their respective memory addresses along with the area of external memory space that has been reserved to implement the Flash memory described in this project.

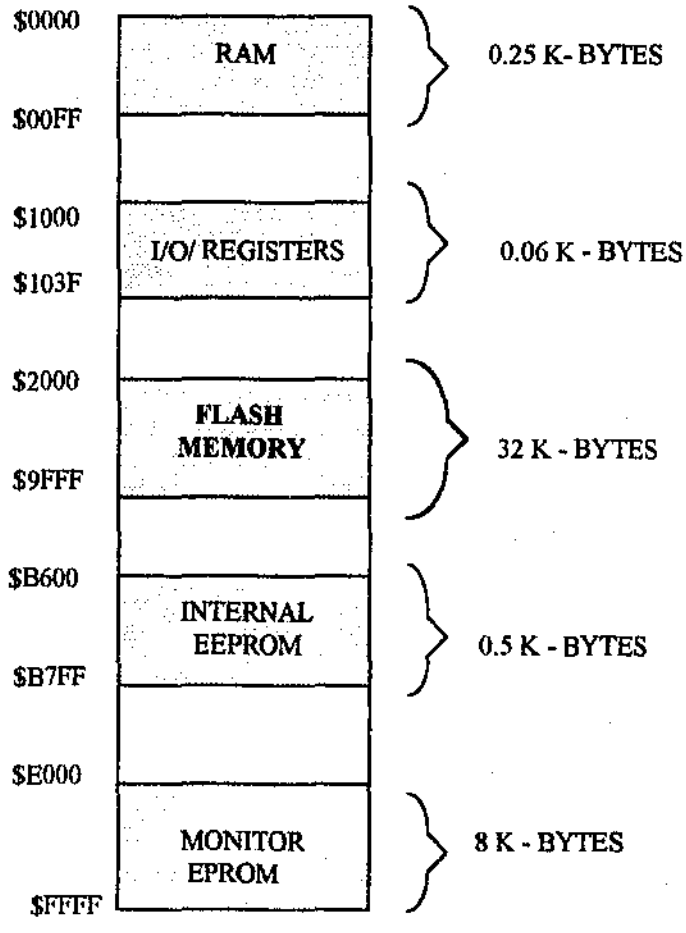


FIG 2 : MC68HC11 MEMORY MAP

The MC68HC11 MCU has a 16 bit address bus which gives a total of 65,536 bytes or 64 kilobytes of addressable space. To extend the memory of the MC68HC11 a portion of the MC68HC11 64 kilobytes of memory must be used to access the Flash memory chips. The logical choice for this portion of memory is the largest memory area that is already available for external use by the user. In this project the area between $2000 and $9FFF restricts the block of external memory to 32 kilobytes which is exactly one quarter of the space provided by the Am29F010 Flash Memory Chip.

Another reason for using this portion of memory is that it does not interfere with the other reserved parts of memory set aside for specific use by the MC68HC11 such as , ROM, RAM, EEPROM, interrupt vectors and I/O registers.

An important factor here that must be noted is that in order to access the full 64 K-Byte address space the MC68HC11 must be allowed to operate in *expanded mode*. Because of this a brief mention of the *expanded mode* operation signals will be mentioned.

SINGLE CHIP MODE

Microcontrollers may work in either single chip mode or expanded mode. In single chip mode the C.P.U. can only access internal memory and I/O modules. All ports are available for I/O data and control signals.

EXPANDED MODE

Sometimes, a microcontroller may requires more memory or I/O ports than are available in the chip itself. If this is the case external data and address line connections will be needed. Typically, some pins can be used either as I/O ports

or as external data and address lines. The MC68HC11 microcontroller can be set up to work either way. How it is set up is called the microcontroller's mode of operation. Because the MC68HC11 requires additional memory circuitry it must operate in the expanded mode for this project.

In this mode the expansion bus is made up of ports B (A15 - A8) and C (A0 - A7): the control signals AS (address strobe) and R/W' (read/write: where " ' " designates a LOW signal). The R/W' allows a read or write operation to or from memory, the AS allows the low-order address and the 8-bit data bus to be multiplexed on the same pins of port C. During the first half of each bus cycle the address information is present along with an AS signal. During the second half of the bus cycle the pins become a bi-directional data bus. AS is an active high latch enable signal for an external address latch. Address information is allowed through the transparent latch while AS is high and is latched when AS drives low.

The address, R/W', and AS signals are active and valid for all bus cycles including accesses to internal memory locations. The E clock is used to enable to external devices to drive data onto the internal data bus during the second half of a read bus cycle (E clock high). The R/W' controls the direction of data transfer.

INTERFACING THE MC68HC11 to FLASH MEMORY.

An interface between MC68HC11 and the Flash memory chips is now considered by utilising the MC68HC11 in expanded mode and allocating the address range $2000 - $9FFF, to incorporate the additional flash memory.

The proposed design ensures that when the MC68HC11 attempts to read or write to address $2000 to $9FFF the MC68HC11 will access a block of the new extended memory incorporating the Flash Memory chips. The following table illustrates the first step in designing of the overall system using Flash Memory. Table 1 overleaf examines the address range of the MC68HC11 used when the Flash Memory chips are required.

Memory address range in Hexadecimal.	Memory address range shown in binary MSB															LSB
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$2000	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
-																
-																
$9FFF	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
	Bits required to select Flash Memory					"Don't care bits"										

TABLE 1 : Bit pattern for address range of MC68HC11 from $2000 to $9FFF

In designing the logic circuits that permits use of the Flash Memory chips, it can be seen that bits 12 and below are of no importance in making the decision whether the Flash Memory is to be used or not : that is to say, the state of bits 12 and below are actually in boolean terminology "don't care states". Note, bits 12 through to bit 0 are used when specifying individual memory locations to be accessed. Below the address $2000 and above the address $9FFF does not warrant use of Flash Memory.

Table 2 below shows the significant bits needed in making the decision whether Flash Memory is to be used by the MC68HC11 or not.

ADDRESS	BITS USED TO SELECT FLASH MEMORY			
	15	14	13	12
$2000	0	0	1	0
-				
-				
-				
$9FFF	0	1	0	1

TABLE 2 : Bit pattern showing significant bits that make a decision on the need for the use of Flash Memory for address range of MC68HC11 from $2000 to $9FFF

Designing a logic circuit that can communicate between the MC68HC11 and the Flash memory chips can be achieved by the use of the Karnaugh map. Figure 3 illustrates the steps in the design using the Karnaugh map that will result in a logic equation to access the Flash Memory chips when the MC68HC11 attempts to read from or write to addresses $2000 to $9FFF.

16- BIT MEMORY ADDRESS	ADDRESS BUS OUTPUTS A15	A14	A13	A12	EXT MEMORY SELECT
0000	0	0	0	0	0
-	0	0	0	1	0
-	0	0	1	0	1
-	0	0	1	1	1
-	0	1	0	0	1
-	0	1	0	1	1
-	0	1	1	0	1
-	0	1	1	1	1
-	1	0	0	0	1
-	1	0	0	1	1
-	1	0	1	0	0
-	1	0	1	1	0
-	1	1	0	0	0
-	1	1	0	1	0
-	1	1	1	0	0
FFFF	1	1	1	1	0

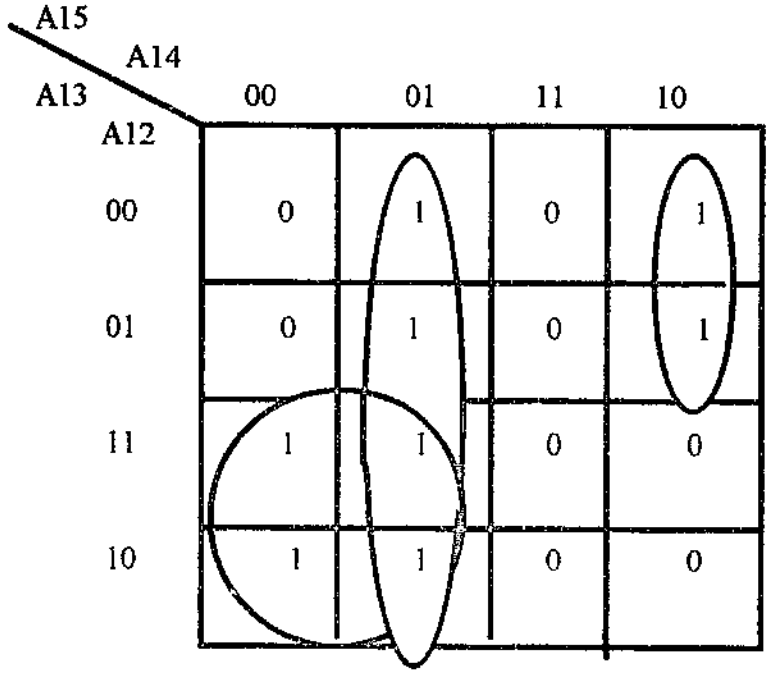


FIG 3 Karnaugh map for design of Flash Memory chip select.

The resulting logic equation is given below and assigned as the Flash Memory Chip Select control signal :-

$$FM = \overline{(A15 * A14) + (A13 * A15) + (A15 * A14 * A13)} \quad \text{..... (1)}$$

To enable access to Flash Memory a chip select signal must be sent to the respective Flash Memory chip , CS' (disscussed later). As the Chip Select signal is low, negation of the above equation is necessary. This now brings us to the following logic equation.

$$FM' = \overline{(A15 * A14) + (A13 * A15) + (A15 * A14 * A13)} \quad \text{..... (2)}$$

Using DeMorgans Theorem this breaks down further to :-

$$FM' = \overline{(A15 + A14) + (A13 + A15) + (A15 + A14 + A13)} \quad \text{..... (3)}$$

Transforming the logic equation into a logic circuit gives the following diagram, FIGURE 4, overleaf.

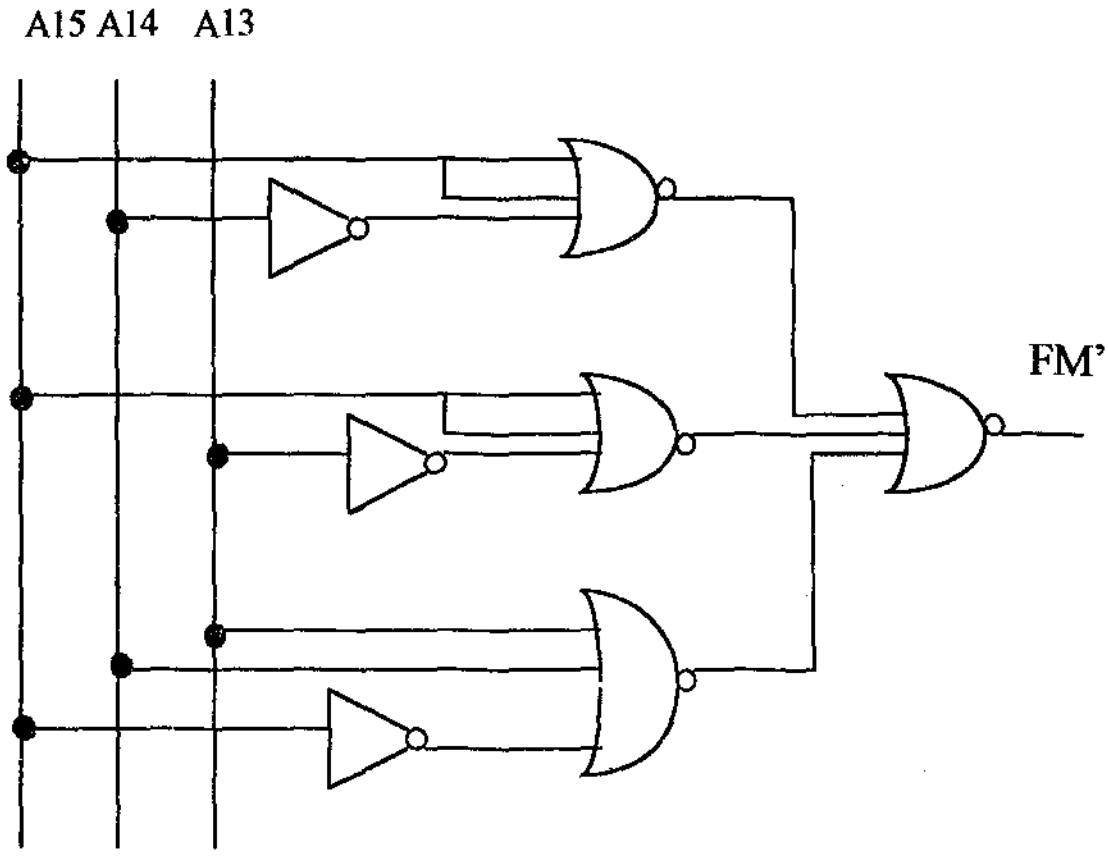


FIG 4. Flash Memory Chip Select Circuit.

The above logic circuit will now be referenced as the "Flash Memory Select" and illustrated as shown in Figure 5 below : Details of schematic outlay is listed in Appendix at end of document. Note also BLOCK diagram of whole circuit of a design is also included in appendix.

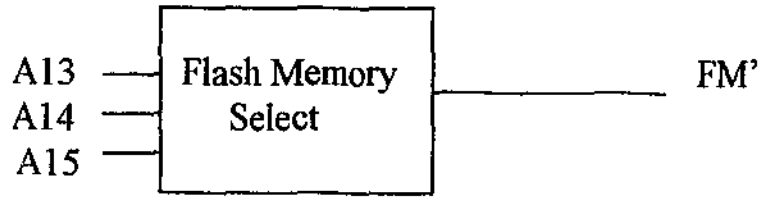


FIG 5. Flash Memory Select Symbol.

CHAPTER 5

DESIGN OF CHOOSING A SPECIFIC FLASH MEMORY CHIP :

After having completed the Flash Memory Chip Select circuitry, a design now needs to be created to choose a specific Flash Memory chip (FMC) from the four employed.

Although there are four Flash Memory chips in the overall design of the interface the microcontroller needs only access one chip at a time to store or read its data. If more than one chip is accessed at the same time when using the write procedure, the same information would be stored at different locations, thus, resulting in a waste of valuable memory space. When reading from the same four chips at the same time a collision of data would occur, the result being error in information. The design must therefore also incorporate the task of accessing only one chip at any one time.

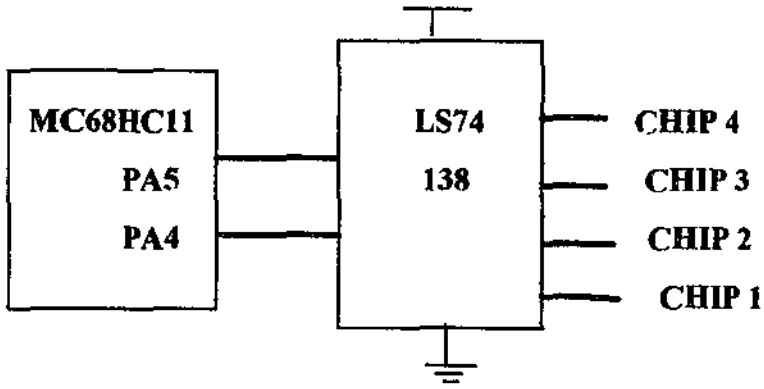


FIG 6: CHOOSING A SPECIFIC FLASH MEMORY CHIP.

THE *CE'* FUNCTION

The secondary storage device that is used in this project is the 32 pin Advanced Micro Devices (AMD) AM29F010 Flash Memory chip (FMC). Pin 22 of this chip incorporates a Chip Enable (CE') function. The function of the CE' is to inform the circuitry in the

chip that when a LOW signal is present at its input, a read or write may occur to or from the chip. If a HIGH signal is present at its input no access to the chip is granted. Because of the CE' function on the Am29F010 FMC, the design incorporating selecting one specific chip out of the four at any one time is made easier. By having only one LOW CE' signal present at any one time on any of the four chips, (while the remaining chips have a HIGH CE' signal), can allow a single specific chip to be selected.

What now needs to be designed is a circuit that has four lines of which one of these can hold a LOW signal while the rest remain HIGH, and the ability to alternate this LOW signal about these four lines. These four lines must also have the disposition to each serve as an input to the CE' on the Flash Memory chip.

In addition to programming specific output control lines on the MC68HC11, using a multiplexed to place the appropriate LOW CE' signal on the Flash Memory chip of choice could serve as possible solution in selecting one of the four chips. However, the chip should only be selected when the microcontroller accesses memory between the range of $2000 and $9FFF. Because of this reason a more than just a multiplexor is required. A more complex design is required.

The Flash Memory chips should only be employed when the Flash Memory Select signal is LOW and a particular Flash Memory chip is chosen. Incorporating the use of a 3-8 Decoder (LS138 chip) and employing a system of four two input OR

gates (74LS32) choice of a specific Flash Memory within the required memory range can be achieved.

To accomplish this the higher Input/Output (I/O) control lines of Port A on the MC68HC11 are used. The I/O control lines (PA5), and (PA4) of Port A are used as input signals to the 74LS138 decoder. When the assigned Ports PA5 and PA4 on the MC68HC11 are programmed to have signals 00, 01, 10, or 11 present at any one time on its I/O lines, then the output of the decoder will ever only have one LOW signal present on of any of its four output lines at any one time. Note that only four of the eight output lines from the decoder are used as there are only four Flash Memory chips used in this project..

By feeding each of the four output lines of the decoder into one of the inputs of each OR gates (74LS32) and by feeding the other input of each the OR gates with the Flash Memory Select signal (FM'), results in only one output of the 74LS32 to have a LOW signal present at any one time. One of each of these four lines from the output of this 74LS32 can then be fed to each CE' pin on the four Flash Memory chips. The circuit to select a specific Flash Memory chip (one of four) within the memory range $2000 - $9FFF has now been created.

Figure 7 overleaf illustrates the pin connection between the 74138 decoder and the 74LS32 OR gates discussed above. It also shows the pin connection between the two integrated circuits and the internal structure of the 74LS32. Note the internal structure of the 74LS138 decoder is omitted as the circuits primary function is to act as a multiplexor.

As this project uses the AMD AM29F010 Flash Memory chip as the primary secondary storage device, a brief description and relevant programming requirements will be mentioned.

Figure 7 below illustrates pin connections between the 74138 decoder and the 74LS32 OR gates. Shown also is the pin connection between the two integrated circuits and the internal structure of the 74LS32.

FIG 7

CHAPTER 6

THE AMD AM29F010 FMC:

The AMD Am29F010 Flash Memory chip (FMC) is a 1 Mbit, 32 pin CMOS memory chip organised as 128K bytes of 8 bits each. It requires 5 volts for programming and or write/erase operations. The Logic symbol is illustrated in Figure 6 below (pin numbers omitted).

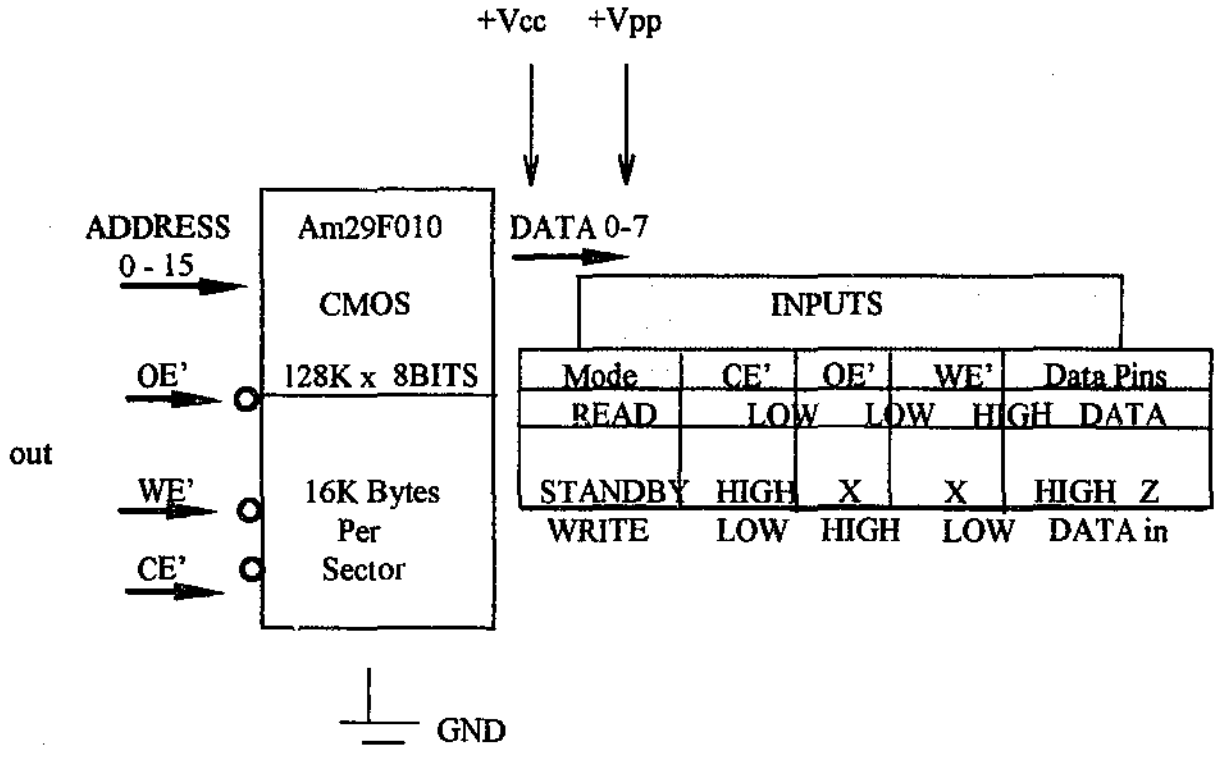


FIG 7. Logic Symbol for the ADM Am29F010 Flash Memory Chip.

READING /WRITING TO THE AMD AM29F1010 FMC

The ADM Am29F1010 FMC has a unique feature and that is of the *command register*. This is a register set aside to manage all of the chip

functions. Command codes are written into this register to control which operations take place inside the chip (e.g., erase, write). These command codes come over the data bus from the microcontroller. State control logic examines the contents of the command register and generates logic and control signals to the rest of the chips circuits to carry out the steps in the operation.

The ADM Am29F010 FMC is programmed by executing the program command sequence. This invokes the embedded program algorithm which is an internal algorithm that automatically times the program pulse widths and verifies cell margin. Each sector can be verified in less than 0.3 seconds. Erase is accomplished by executing the erase command sequence. This will invoke the Embedded Erase algorithm which is an internal algorithm that automatically preprograms the array if it is not already programmed before executing the erase operation. The entire memory is typically erased and verified in three seconds.

The ADM Am29F010 FMC has an access times between 45ns and 120ns. Eliminating bus contention the device has three control inputs - CE' (Chip Enable), OE' (Output Enable), and WE' (Write Enable). CE' allows the chip to be accessed, the OE' allows the chip to be read from and the WE' allows the chip to be written to.

The ADM Am29F010 FMC technology use Fowler-Nordhiem tunneling to electrically erase the entire chip or all bits within the sector. The bytes are programmed one byte at a time using the EPROM programming mechanism of hot electron injection.

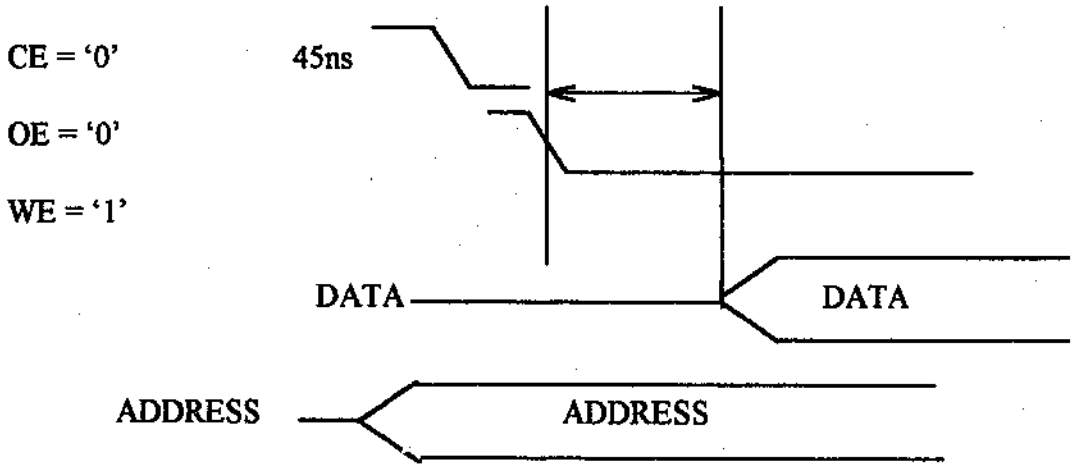
AM29F010 READ COMMAND :

During power up the ADM Am29F010 automatically sets the internal state machine into READ mode, this is a unique feature that assists in data protection. and will be discussed in detail under the DATA PROTECTION section.

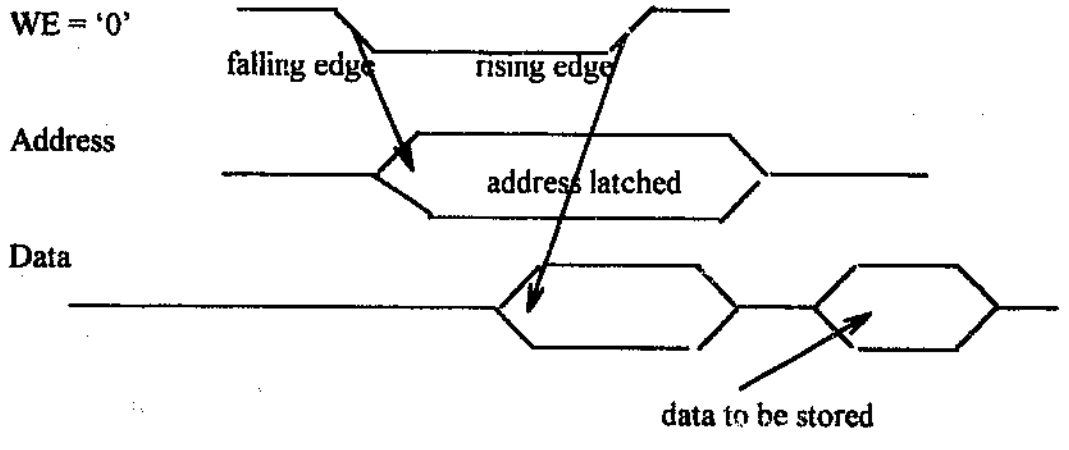
To set the chip up for a read operation, it is first necessary to set the command registers to all 0's (00_{16}). This can be done by applying 00_{16} to the data pins and pulsing the WE' signal LOW', CE = LOW and OE = HIGH. Once this achieved device is set up to have data read from the 32K-Byte memory matrix. Memory is read in the following way and is listed overleaf.

1. Apply the address to be read from.
2. Set WE = HIGH.
3. Set CE = LOW
4. Set OE = LOW.

READ MODE



WRITE MODE

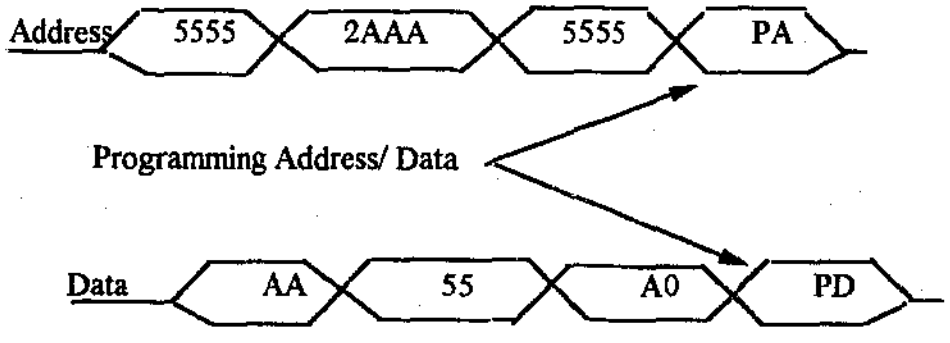


WRITE MODE :

The programming and erasure of data are accomplished as mentioned earlier via the command register. The command register is used to store the commands along with the address and data information needed to execute the command. To write to the command register requires the following control signals. It should be noted here that standard microprocessor write timings are used

1. WE = LOW.
2. CE = LOW
3. OE = HIGH.

The device is programmed on a byte-by-byte ba.is using a four bus cycle operation. The four bus cycle operation comprises of : two "unlock" write cycles, a program "set-up" command and a data write cycle.



Addresses are latched on the falling edge of the WE pulse, while data is latched on the rising edge of the WE pulse. No other signals are required.

ERASE COMMAND

There are two types of Erase operations.

1. Chip Erase and
2. Sector Erase.

CHIP ERASE :

Chip erase is a six bus cycle operation. There are two "unlock" write cycles followed by the "set-up" command, two more "unlock" write cycles, then the chip erase command.

The erase begins on the rising edge of the last WE pulse in the command sequence and terminates when the data on DQ7 is a "1". The device then returns to the read status.

SECTOR ERASE:

Sector erase is a six bus cycle operation. There are two "unlock" write cycles followed by the "set-up" command, two more "unlock" write cycles, then the sector erase command.

The sector address (any address within the desired sector) is latched on the falling edge of the WE, while the command data is latched on the rising edge of the WE. A time-out of 80us from the rising edge of the last sector erase command will initiate the sector erase command. It should be noted here that multiple sector erase's can be done concurrently.

DATA POLLING

Data polling is a method used to indicate to the host that the embedded algorithms are in progress or have been completed. During an embedded programming algorithm, complementary data will be read from DQ_7. On completion of a programmed write operation DQ_7 will produce the true data.

During an Embedded Erase Algorithm :
DQ_7 = '0' during the erase operation.
DQ_7 = '1' on completion.

Toggle bit (DQ_6)

The toggle bit is used in a similar way to the DQ_7. It is a method used to indicate to the host that the Embedded Algorithms are in progress or have been completed. DQ_6 toggle's between '1' and '0' whilst a process is taking place. DQ_6 shows valid data on completion of a process on successive attempts to read this bit.

DQ5 Exceeded Time Limit

DQ5 will indicate if the program or erase time has exceeded the specified limits (internal pulse count). When this happens DQ5 = '1'. When this happens a failure has occurred indicating that the program or erase cycle was not successfully completed. If this failure occurs during the sector erase operation , it specifies that the particular sector is bad and cannot be reused. However, other sectors are still functional and may be used for additional program or erase operations.

DQ4 HARDWARE SEQUENCE FLAG

When DQ5 is set to '1', then DQ4 will indicate whether a programming (DQ4 = '0') or an Erase (DQ4 = '1') caused the fault.

DQ3 SECTOR ERASE TIMER

After completion of the initial sector erase command sequence the sector erase time-out will begin. The DQ3 bit will remain low '0' until the time-out is complete. On completion of a time-out DQ3 will be set to a '1'.

DATA PROTECTION

The AMD Am29F0101 is designed to offer protection against accidental eraser or programming caused by spurious system level signals that may exist during power transitions. Another unique feature that assists in data protection is during power up. When the system is powered up the AMD Am29F0101 automatically sets the internal state machine into READ mode. This prevents any alteration of the memory contents

As a brief descroption of the ADM Am29F010 and its basic programming capabilities has been given and access to specific ADM Am29F010 chips has now been designed, access to specific addresses in the ADM Am29F010 FMC now needs to be created.

CHAPTER 7

ACCESSING SPECIFIC ADDRESSES IN THE ADM AM29F010 FMC

The ADM Am29F010 FMC has 17 address lines, A0 - A16. The address range in the ADM Am29F010 FMC is 00000h to 1FFFFh, which gives a total of 131 072 single addresses. The MC68HC11 has 16 address lines available, A0 - A15. Eight address lines of Port B plus eight address lines of Port C. This gives a total of 65 536 (2^{16}) or 10000h addresses that be accessed. As the ADM Am29F010 FMC has a greater number of addresses than the MC68HC11 can access using address lines A0 - A15, a way must be found so the full address range of the ADM Am29F010 FMC can be accessed using the address lines available by the MC68HC11.

One solution to access all address spaces of the ADM Am29F010 FMC by the MC68HC11, is to break the ADM Am29F010 FMC's memory architecture into four sectors and controlling the input values A16 and A15 of the ADM Am29F010 FMC.

By using these values of bits A15 and A16 of the ADM Am29F010 FMC's its memory architecture could be broken up into a maximum of four sectors. For instance if A16 and A15 were assigned the values 00, it would be in sector 1. If A16 and A15 were assigned the values 01, it would be sector 2. If A16 and A15 were assigned the values 10, it would be sector 3, and if A16 and A15 were assigned the values 11 it would be sector 4. As A16 and A15 are the MSB's of the ADM Am29F010 FMC's address the value of A16 and A15, chooses which sector the address falls in and the range of addresses within that sector that can be accessed. Figure 8 below illustrates this further:

SECTOR	A16	A15	ADDRESS RANGE
1	0	0	0000 - 32 768
2	0	1	32 769 - 65 536
3	1	0	65 537 - 98304
4	1	1	98305 - 131072

FIG 8: Sector /Address range of the Am29F010 FMC

If A16 and A15 were chosen as 10, then the only addresses that could be accessed would be addresses from 65537 to 98304. Any other value of A16 and A15 would change the sector and thus the address range.

If the address architecture of the ADM Am29F010 FMC is arranged in this way the MC68HC11 can access all address spaces.

Bits A16 and A15 of the ADM Am29F010 FMC can be controlled by connecting them to ports PD3 and PD2 of the MC68HC11 and programming their values from 1- 4.

As four sectors is sufficient for accessing the complete of the ADM Am29F010 FMC dividing it up into smaller sectors would be possible as the address range needs to include addresses 2AAA and 5555 which go to the command register in the Am29F010 chip when erasing or writing processes take place. It would also tie up more hardware of the MC68HC11 as more lines will be needed to access more blocks. This will be more clearly illustrated when describing Sector Access under the following heading "SECTOR SELECT OF THE ADM Am29F010 FMC".

SECTOR SELECT OF THE ADM AM29F010 FMC.

As the architecture of the ADM Am29F010 FMC is now made up four sectors the next task at hand is to devise a way of accessing these sectors. The ADM Am29F010 FMC.

To accomplish this two of the connector lines of Port D of the MC68HC11 are used. PD3 and PD4 of Port D are used. As binary operates in powers of two, to choose any of four different sectors two bits are needed. 00 = 1, 01 = 2, 10 = 3 and 11 = 4. When the assigned Port connectors PD3 and PD2 on the MC68HC11 are programmed to have signals 00, 01, 10, or 11 any of the four blocks can be chosen.

The choice of sector can be accomplished by programming the MC68HC11 with the binary values above, but strategic placement of these signals are important in obtaining the correct addressing on the ADM Am29F010 FMC. By using the binary values from ports PD3 and PD2 as inputs to the ADM Am29F010 FMC A16 (pin 2) and A15 (pin 3) the correct access to the wanted sector and wanted address within that sector is easily achieved.

The maximum address of the ADM Am29F010 FMC is 131 072. Decrementing this is stages of 32 768 (2^15), which is the decided maximum address value of the sector, gives four levels before the address value of 0 is reached. Figure 9 overleaf illustrates the interconnection of the MC68HC11 and the circuits for interfacing the Flash Memory :

CHAPTER 8

READ/WRITE

The MC68HC11 sends its READ/WRITE (R/W') signal in conjunction with the on board clock signal E. To Read, a HIGH signal is output. To WRITE a LOW signal is output. The ADM Am29F010 FMC however, has to have either control lines OE' to READ and WE' LOW to WRITE. This poses a problem as both of these signals of the ADM Am29F010 FMC need to be LOW to READ or WRITE and the MC68HC11 R/W signals have opposite logic to each other. Therefore a straight connection from the MC68HC11 READ output to the ADM Am29F010 FMC Output Enable and a straight connection from the MC68HC11 WRITE output to the ADM Am29F010 FMC Write Enable would not work when reading to or writing from.

A design has to be created to turn the HIGH READ signal of the MC68HC11 to a LOW signal (to Read From the ADM Am29F010 FMC, OE') and keep the WRITE signal of the MC68HC11 the same when writing to the ADM Am29F010 FMC.

Incorporating the E clock signal from the MC68HC11 a logic circuit consisting of several NAND gates (LS7400) enables the HIGH READ signal of the MC68HC11 to be converted to a LOW signal when reading from the ADM Am29F010 FMC (OE') and the WRITE signal of the MC68HC11 can be kept the same when writing to the ADM Am29F010 FMC (WE').

The circuit to perform this function is shown below in Figure 10 (pin connections listed).

FIGURE 10 belows shows the schematic pin connections of the READ/WRITE using the LS 7400 chip.

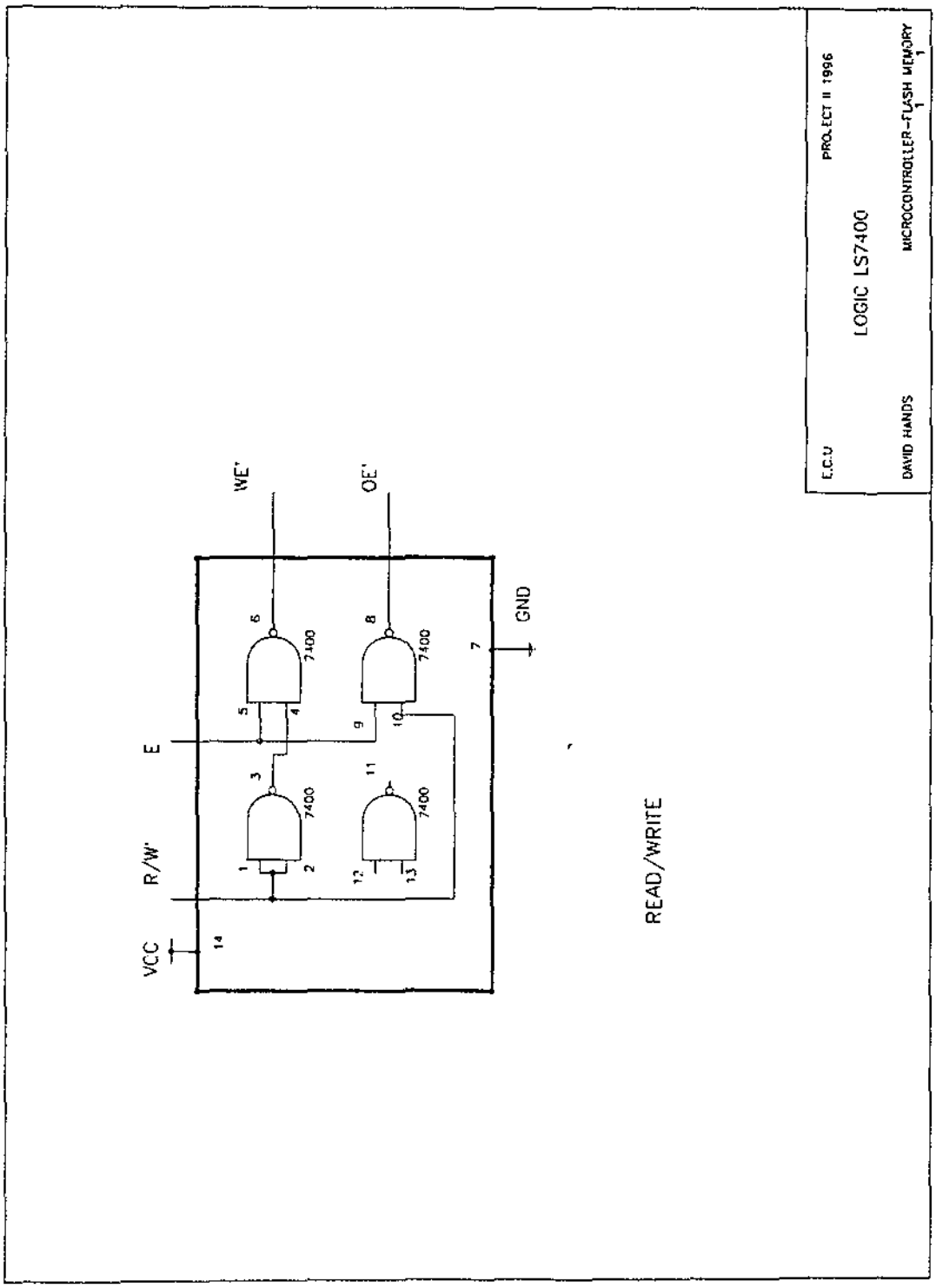
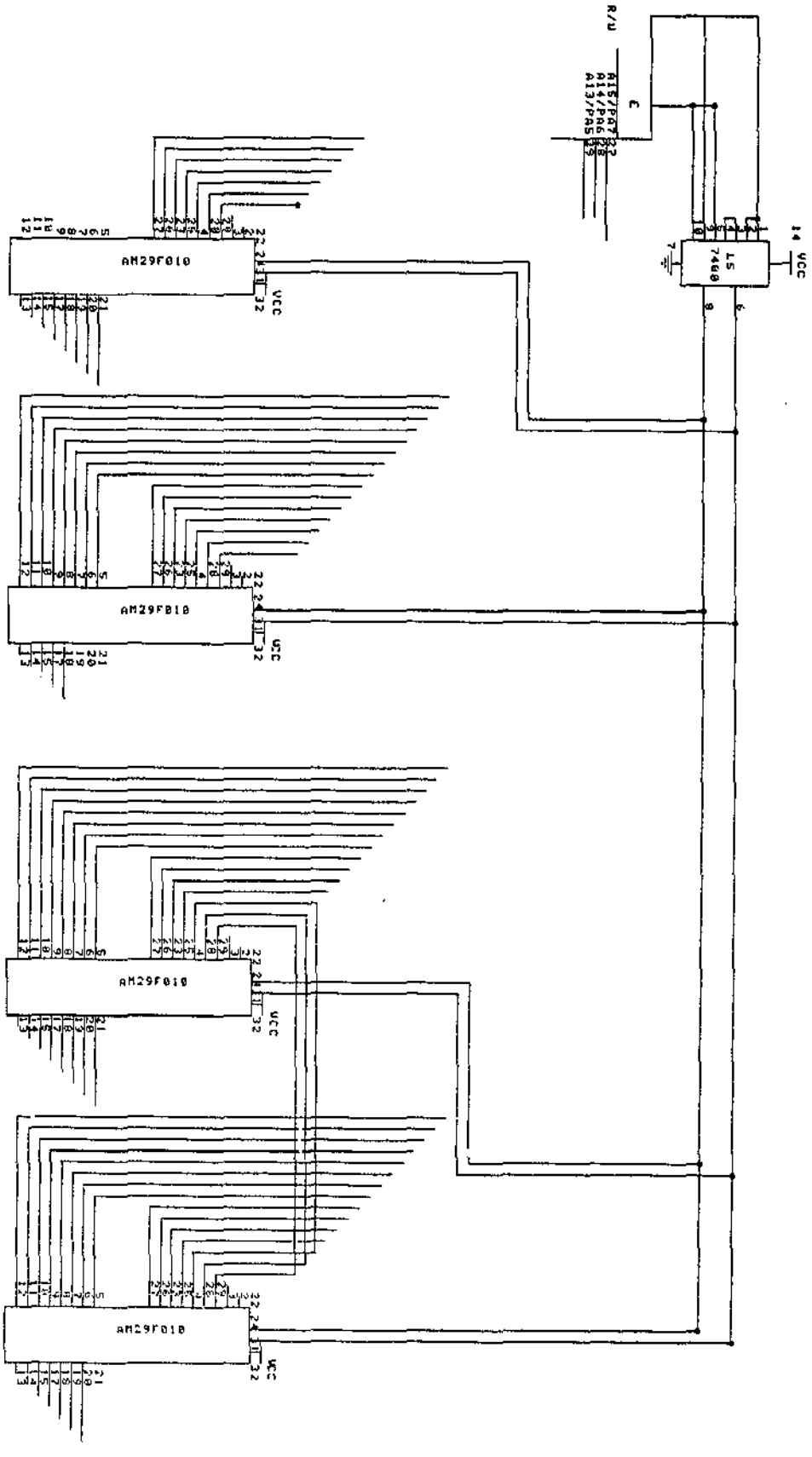


FIG 10


R/W
E
A15/PA7
A14/PA6
A13/PA5
AM29F010
AM29F010
AM29F010
AM29F010
VCC
74LS08

The assigned WE' (pin 6) and OE' (pin 8) output pins of the LS7400 are connected to all WE' (pin 31) and OE'(pin 24) pins of the ADM Am29F010 FMC.

ADDRESS/DATA SEPERATION

A LS74373 latch is used in interfacing the MC68HC11 to the ADM Am29F010 FMC. The function of this latch is to separate the address and data signal into their respective bus paths.

The 74LS373 latch demultiplexex Port C of the MC68HC11 so the ADM Am29F010 FMC receives the lower byte of the MC68HC11 for the entire clock cycle. The address strobe of the MC68HC11 (AS) drives the latch enable input during the first half of the clock cycle, this latches the low byte address. During the second half of the clock cycle the data is placed onto Port C which also placed at the data inputs of the ADM Am29F010 FMC; Port C being directly hard wired to the input data pins of the Figure 11 show Port C and Port B's connection to the ADM Am29F010 FMC data inputs.

The design of the interface from the MC68HC11 to the AMD AmF0101 has now been designed. An additional stage of this project has also been to construct the hardware of the above mentioned and test it accordingly. The following chapter relates to the testing of the interface hardware of the MC68HC11 to the four AmF0101 Flash Memory Chips.

CHAPTER 9

WIRE-WRAPPING

Although not incorporated directly with the design of the extended secondary storage, wire-wrapping plays a considerable part in connecting the interface to the MC68HC11. Due to the importance of these connections, a brief description of the wire-wrapping procedure is mentioned.

With the testing of the MC68HC11 microcontroller on the EVB verified as correct and in working order, a decision on the method of interconnection of the various semi-conductor chips to one another and to the Mc68HC11 needs to be established. The choice of this interconnection between these components was that of the Wire-Wrapping method. The reason for using this method was that Wire-Wrapping is simple to use, fast and easy to rectify (in the advent of the wrong pin being incorrectly connected). This method also eliminates the need for soldering and is less expensive as no P.C.B design and construction is required. The tools required for Wire-Wrapping are;

- Wire-Wrap tool for the wrap-end ,
- Wire-Wrap tool for the de-wrap end and a
- Wire stripper.

CONNECTING TERMINALS OR PINS

Before any connection is done the wire needs to be measured from the point of origin to its destination. The length of the wire should allow for its placement length around other chips and components (i.e. not just the shortest route) and a certain amount of

unrestrained wire (slack). Although some users of wire-wrap would argue that there is no need to allow for this extra relaxed wire, in certain applications or under various conditions the slack is a must. It avoids the possibility that at some stage the wire could stretch and hence would break the conductor. For instance, in applications such as Air-Craft microcontroller use where there is a large amount of vibration

After the appropriate length of wire is measured it should then be cut to size. The wire should then be stripped at the ends by approximately 2 cm (this ensures a good and clean contact is made with the wire-wrap pin). The next step is to insert the stripped wire into the wrap-hold part of the wrap end. Once this is done, the guide hole of the wrap-end together with the wire should be inserted into the relevant pin. The insulated part of the wire should then be firmly held (this avoids sucking in the insulated wire around the wire-wrap pin) and twisted carefully with speed until all the wire in the anvil is wrapped around the pin. The pin should have approximately three wraps of wire insulation and five wraps of clean conductor. The same process is then repeated for all pins that require wire-wrapping. After each wire has been connected the associated line on the circuit diagram is marked (highlighted).

To check if a connection has been established, a multimeter with a continuity beeper should be used. Failing that type of instrument, a check of resistance should be undertaken. A resistance reading of infinity would mean a definite open circuit. A reading of 0-2 ohms would be considered as a successful connection.

POTENTIAL PROBLEMS.

- Due to the slim width of the wire, the conductor when being stripped or by some other cause can brake inside the insulation and go undetected by the naked eye. A continuity test should be undertaken before placement of any wire.

- The placement between the Wire-Wrap pins is very close. Protruding ends of stripped wire may come into contact with a pin without intention if the wire

- wrapping is not done properly. This is usually invisible to the naked eye so a continuity test of surrounding pins should be undertaken.

- The stripped wire can often break if the wire-wrapping is not done properly or with wire that's been wire-wrapped before. As a result broken pieces of wire between pins can be quite common, especially if more than a few wires are wire-wrapped. In most cases this is unavoidable but an easy remedy is just to tap any loose bits free.

- If de-wrapped wire is re-used to wire-wrap, the wire in the anvil can sometimes snap. When this occurs it is usually hard to detect, the insulated wire is wrapped giving the illusion of a connection. The wire is connected, but there is no connection of the conductor. In most cases this is usually picked up by the continuity test which should be undertaken after all placement of wire.

Overall, the advantages of Wire-Wrapping for the development work usually outweighs any disadvantage.

All connections between pins, connectors and terminals were done using wire wrap.

CHAPTER 10

TESTING OF LOGIC GATES ON THE 68HC11:

OUTLAY OF MC68HC11 BOARD WITH LOGIC GATES:

As already established wire-wrapping is of key importance in the implementation of the MC68HC11-Flash Memory project. An important feature that needs to be noted before any wire-wrapping is undertaken is that of the logic gate layout on the MC68HC11 EVB. Viewing the logic gates from a top view is the inverse of viewing the logic gates from a bottom view, because of this inverse of pin configuration it is imperative that this is taken into account before any wire-wrapping is undertaken. By noting the composition of the semi conductor chips viewed from both top and underneath the MC68HC11 EVB dramatically reduces the chance of making any mistakes when wire wrapping and is a fundamental exercise in the organisation of the design of the Microcontroller-Flash Memory project. Illustrated below in Figure 12 and 13 is the top and bottom view of the MC68HC11 EVB.

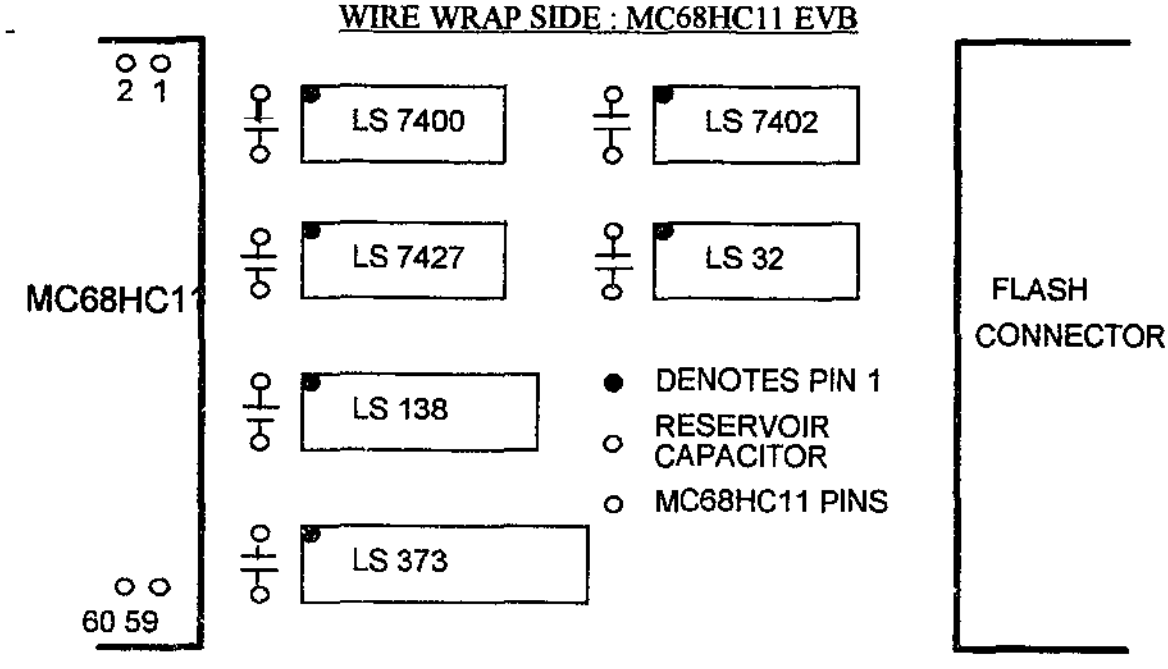


FIG 12: Bottom View of MC68HC11 EVB logic gate layout

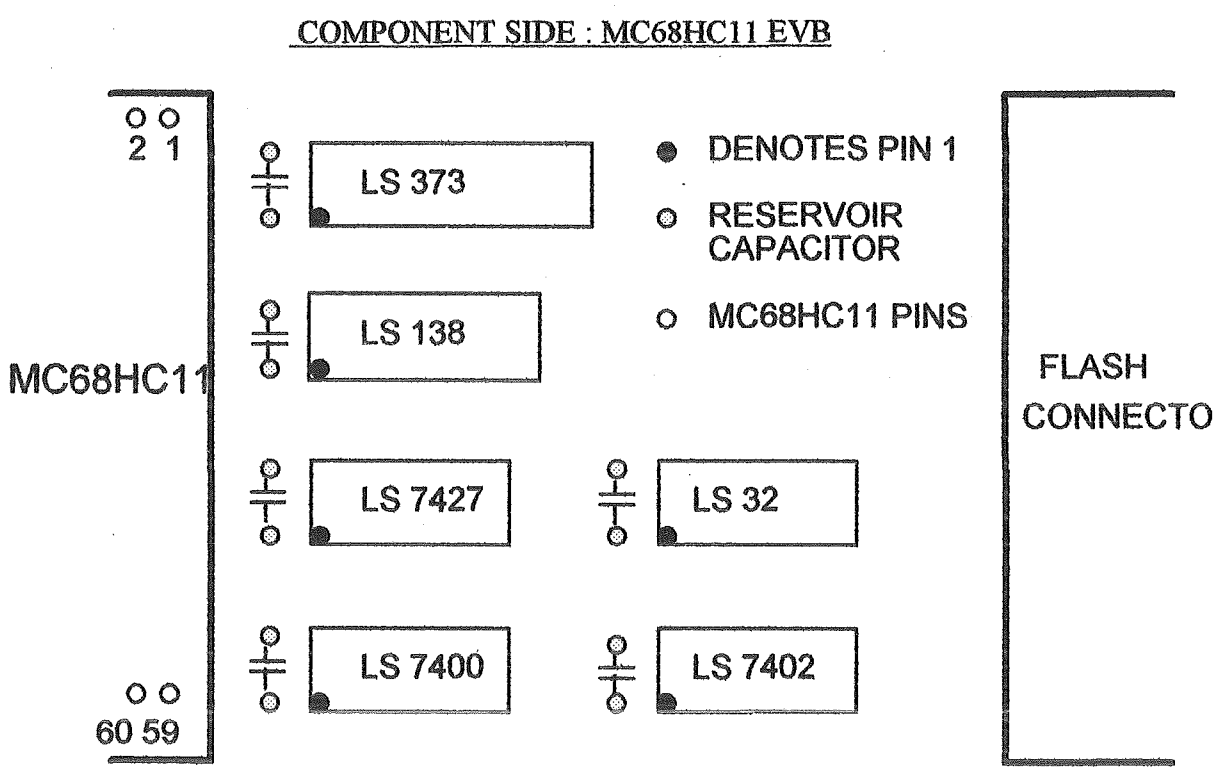
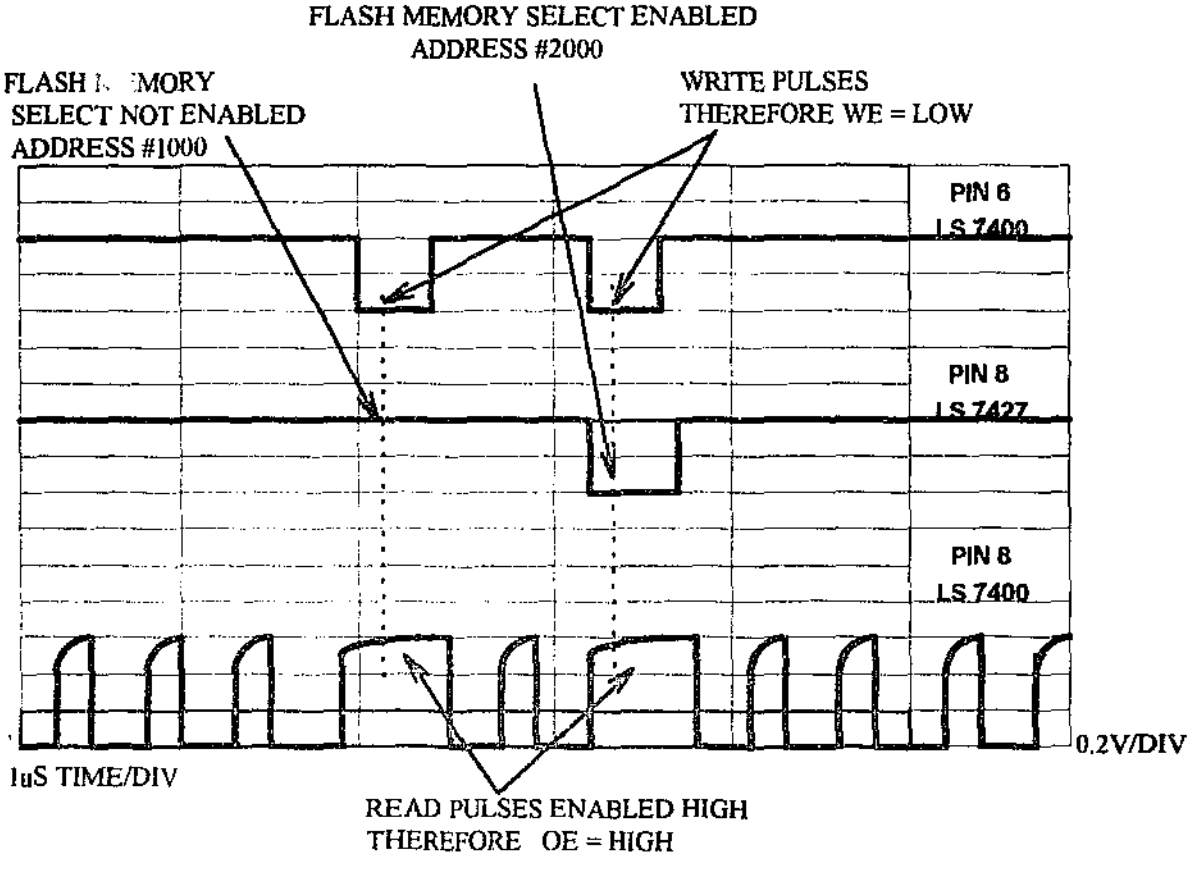


FIG 13 : Top View of MC68HC11 EVB logic gate layout

TESTING OF LOGIC GATES ON THE MC68HC11:

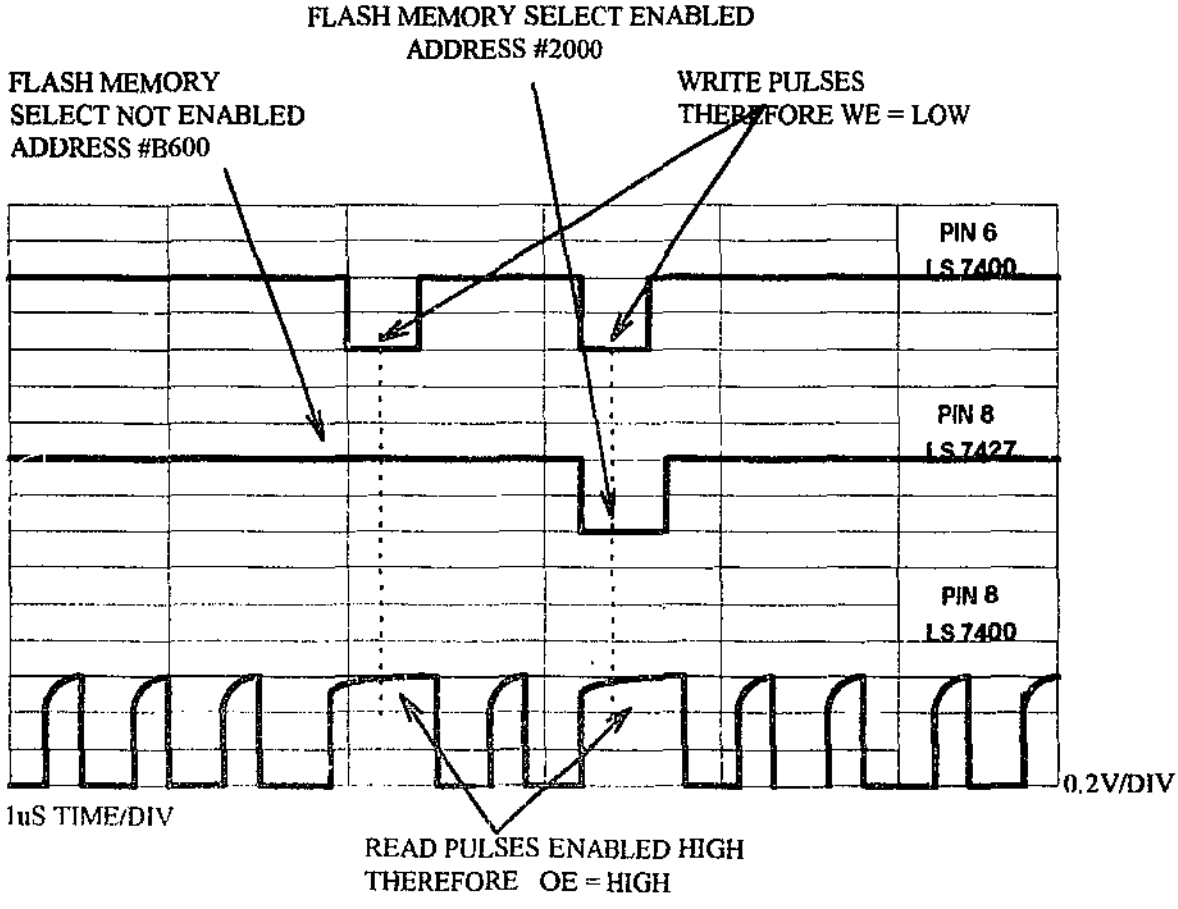
Program: Testing of LS7400, LS7402, LS7427: Testing store instructions for below range of FMS and in range of FMS.

MEMORY LOCATION	INSTRUCTION	COMMENTS
0100	staa 1000	;writes value to Ports B & C outside of range of Flash Memory Select.
0103	staa 2000	;writes value to Ports B & C inside the range of Flash Memory Select.
0106	nop	;no operation, used to help distinguish signals
0107	nop	
0108	nop	
0109	bra 100	;branches back to 100, Infinite loop
010B	wai	;end of program.



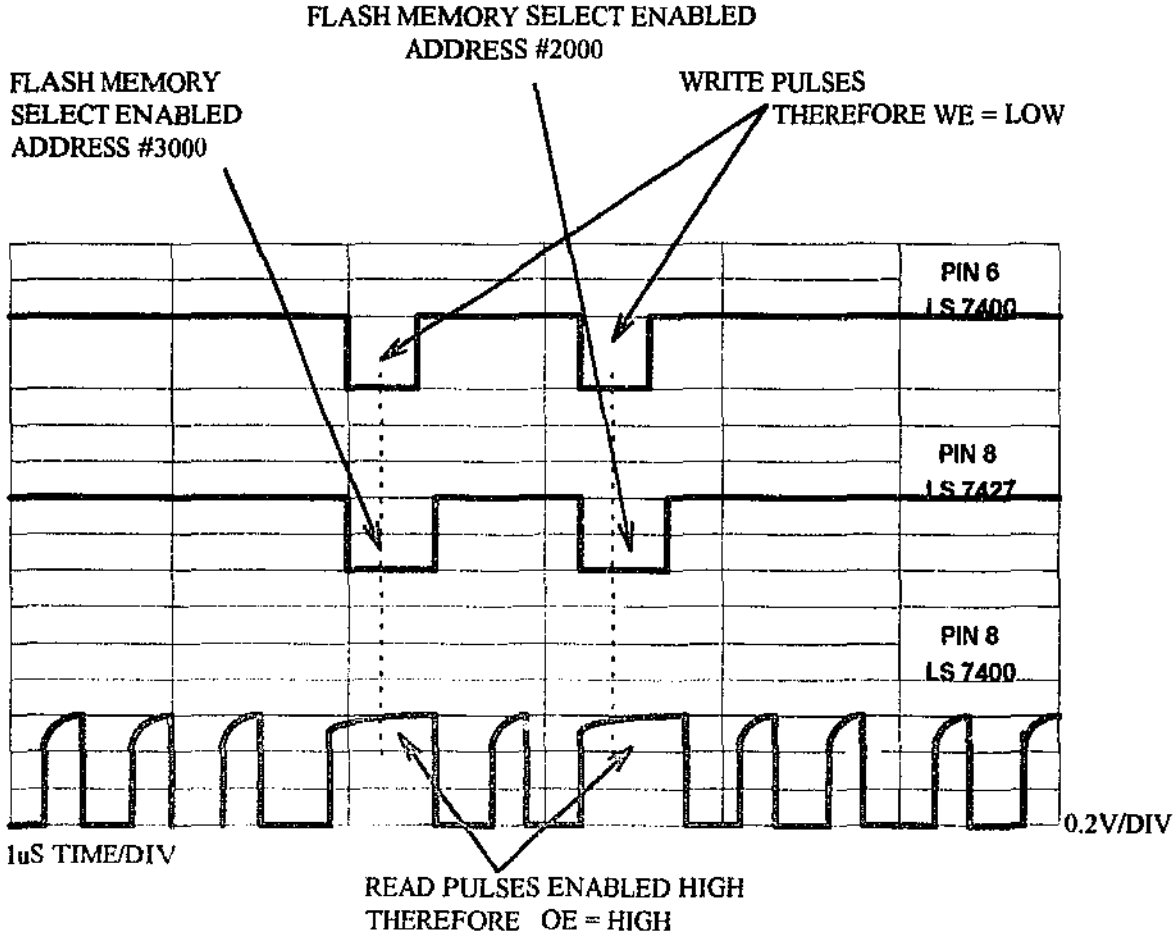
Program: : Testing of LS7400, LS7402, LS7427: Testing store instructions for above the range of FMS and in range of FMS.

MEMORY LOCATION	INSTRUCTION	COMMENTS
0100	staa B600	;writes value to Ports B & C outside of range of Flash Memory Select.
0103	staa 2000	;writes value to Ports B & C inside the range of Flash Memory Select.
0106	nop	;no operation, used to help distinguish signals
0107	nop	
0108	nop	
0109	bra 100	;branches back to 100, Infinite loop
010B	wai	;end of program.



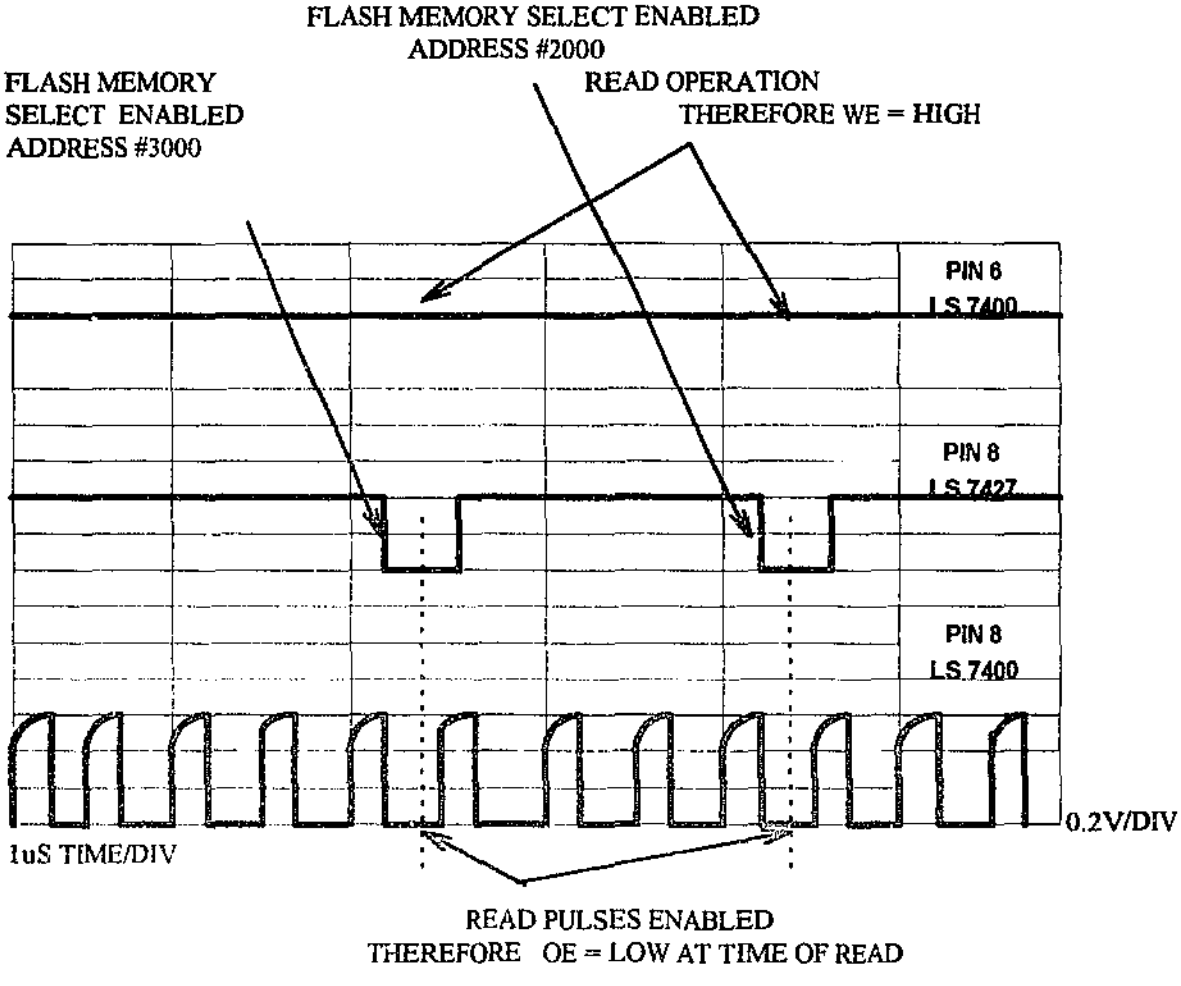
Program: : Testing of LS7400, LS7402, LS7427: Testing store instructions in range of FMS.

MEMORY LOCATION	INSTRUCTION	COMMENTS
0100	staa 3000	;writes value to Ports B & C inside of range of Flash Memory Select.
0103	staa 2000	;writes value to Ports B & C inside the range of Flash Memory Select.
0106	nop	;no operation, used to help distinguish signals
0107	nop	
0108	nop	
0109	bra 100	;branches back to 100, Infinite loop
010B	wai	;end of program.



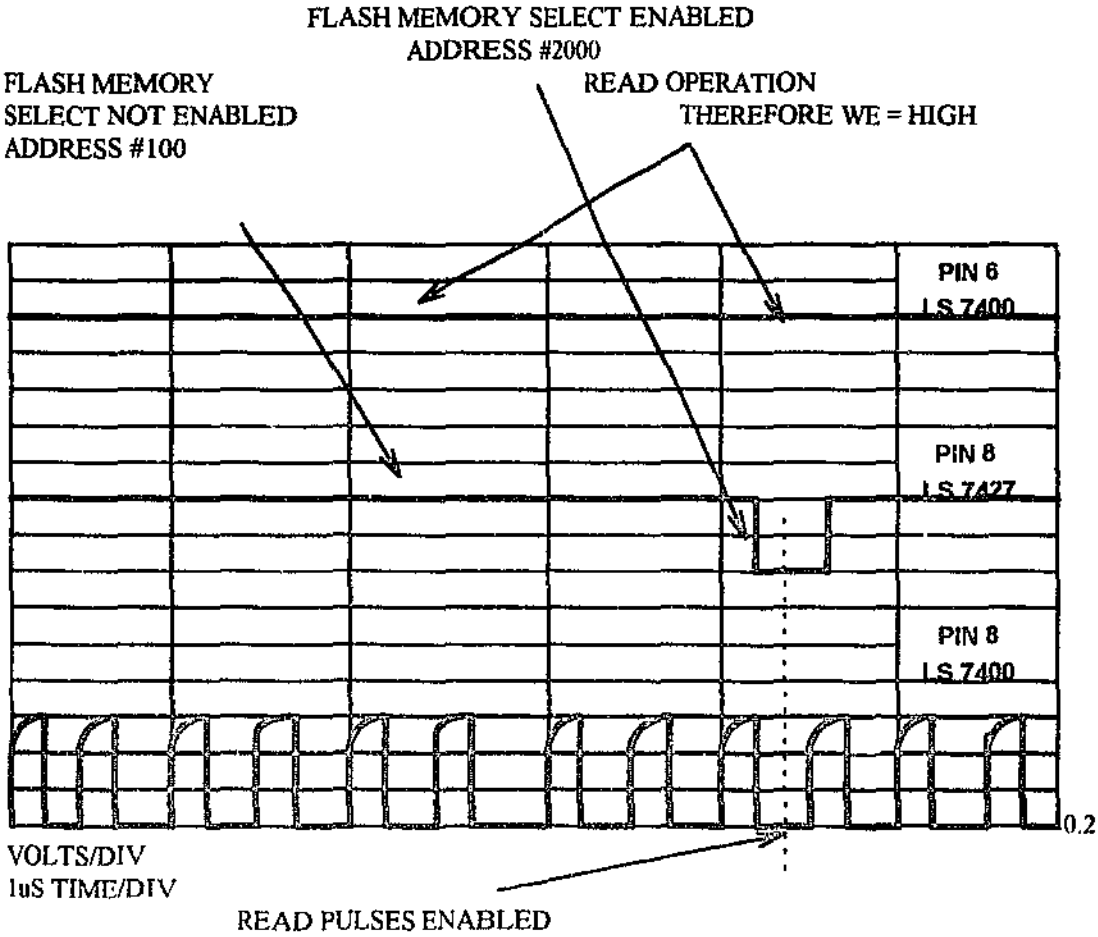
Program : Testing of LS7400, LS7402, LS7427: Testing load instructions in range of FMS.

MEMORY LOCATION	INSTRUCTION	COMMENTS
0100	ldaa 3000	;loads value to Ports B & C inside range of Flash Memory Select.
0103	ldaa 2000	;loads another value to Ports B & C inside the range of Flash Memory Select.
0106	nop	;no operation, used to help distinguish signals
0107	nop	
0108	nop	
0109	bra 100	;branches back to 100, Infinite loop
010B	wai	;end of program.



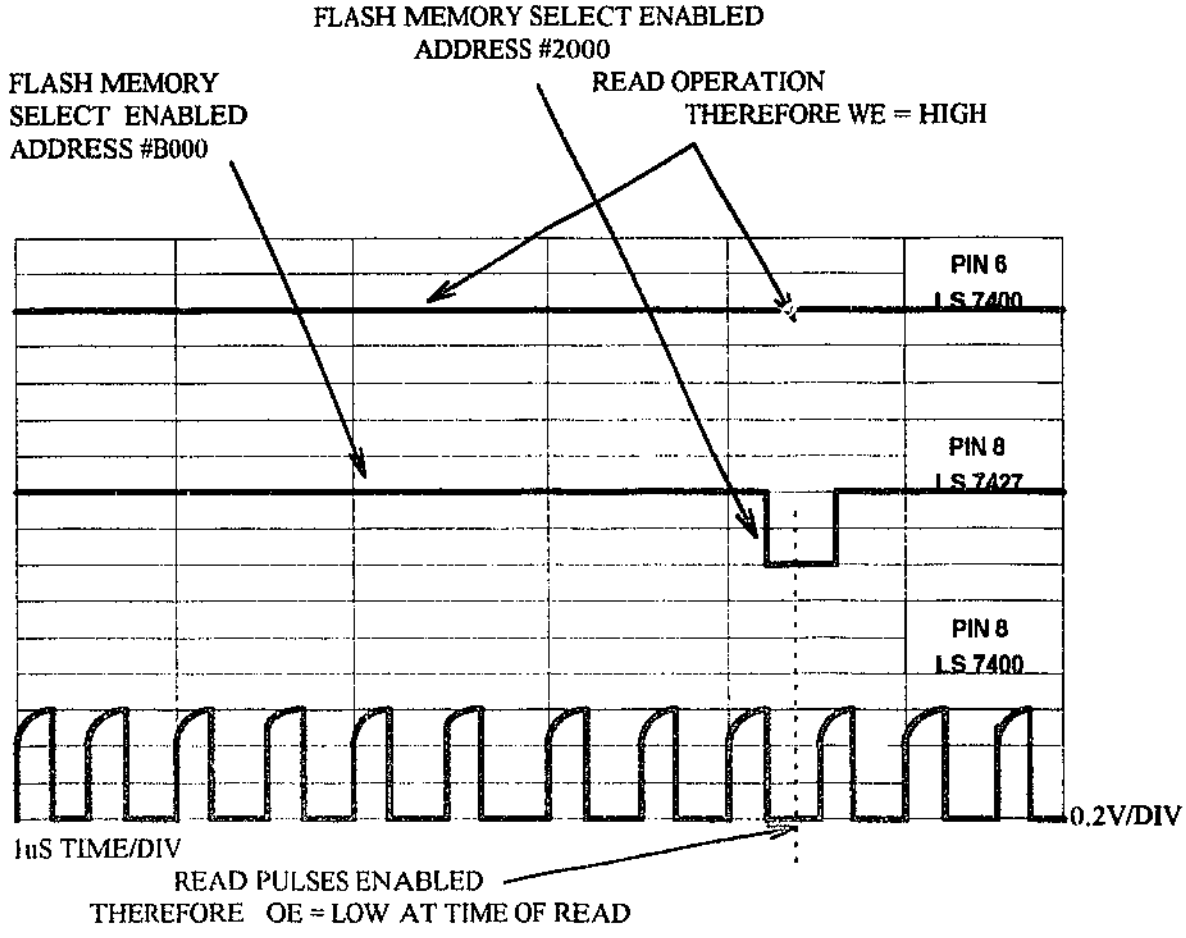
Program : Testing of LS7400, LS7402, LS7427: Testing load instructions for below range of FMS and in range of FMS.

MEMORY LOCATION	INSTRUCTION	COMMENTS
0100	ldaa 100	;loads value to Ports B & C outside range of Flash Memory Select.
0103	ldaa 2000	;loads another value to Ports B & C inside the range of Flash Memory Select.
0106	nop	;no operation, used to help distinguish signals
0107	nop	
0108	nop	
0109	bra 100	;branches back to 100, Infinite loop
010B	wai	;end of program.



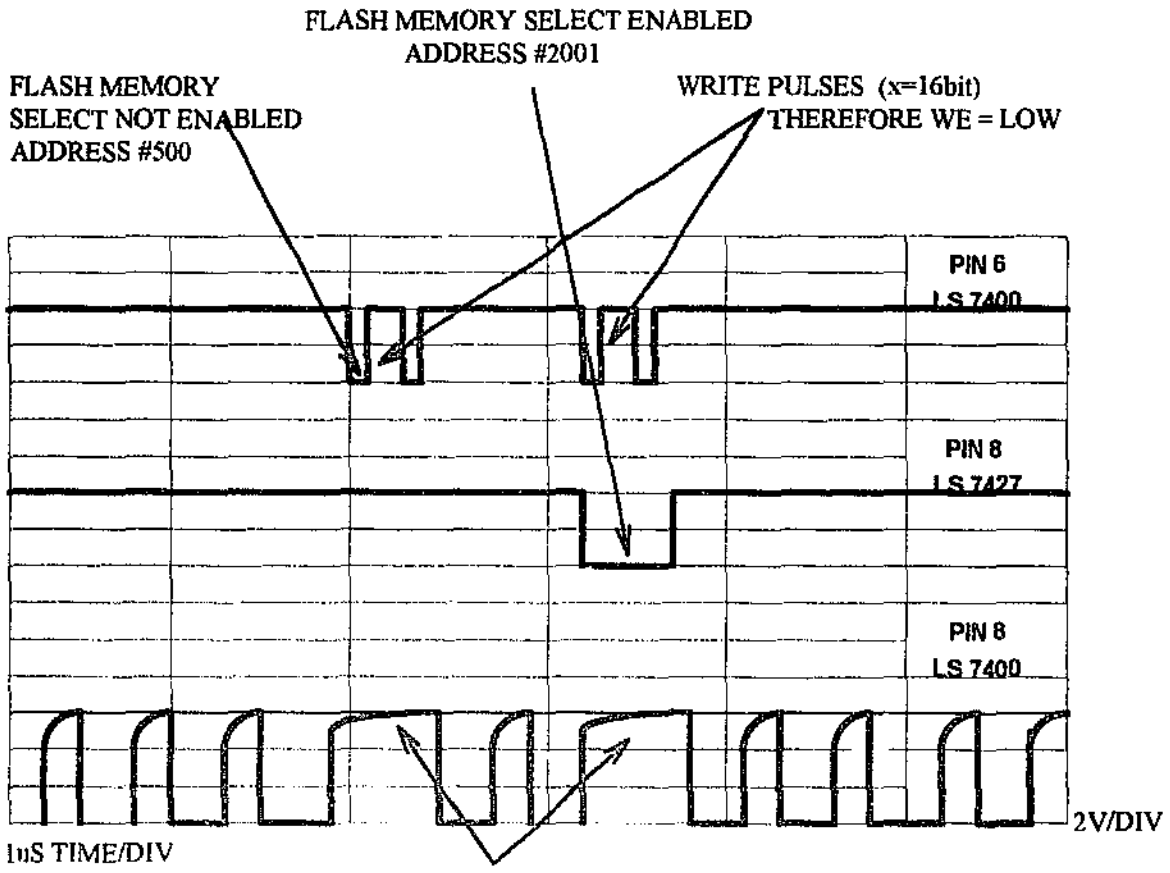
Program : Testing of LS7400, LS7402, LS7427: Testing load instructions for above range of FMS and in range of FMS.

MEMORY LOCATION	INSTRUCTION	COMMENTS
0100	ldaa B000	;loads value to Ports B & C outside range of Flash Memory Select.
0103	ldaa 2000	;loads another value to Ports B & C inside the range of Flash Memory Select.
0106	nop	;no operation, used to help distinguish signals
0107	nop	
0108	nop	
0109	bra 100	;branches back to 100, Infinite loop
010B	wai	;end of program.



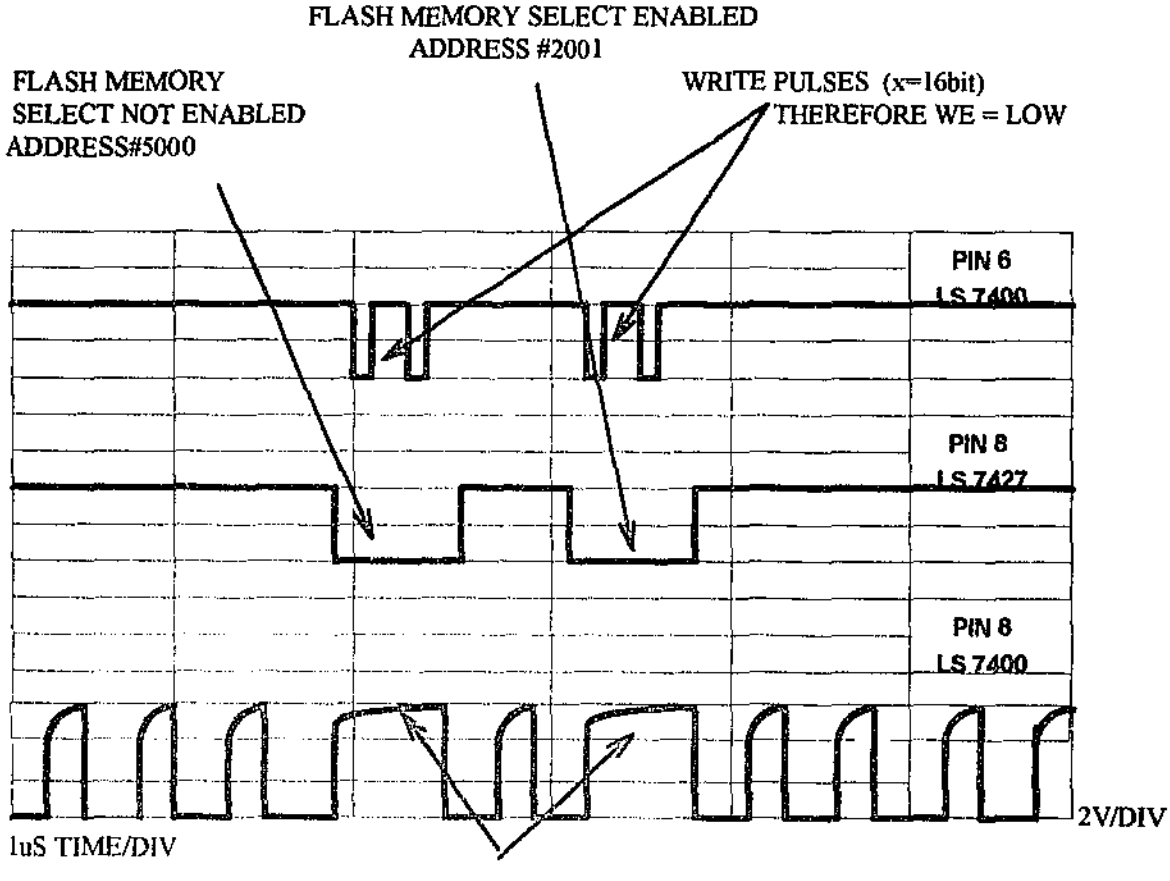
Program: : Testing of LS7400, LS7402, LS7427: Testing store instructions in range of FMS.

MEMORY LOCATION	INSTRUCTION	COMMENTS
0100	stx 500	;writes value to Ports B & C outside of range of Flash Memory Select.
0103	stx 2001	;writes value to Ports B & C inside the range of Flash Memory Select.
0106	nop	;no operation, used to help distinguish signals
0107	nop	
0108	nop	
0109	bra 100	;branches back to 100, Infinite loop
010B	wai	;end of program.



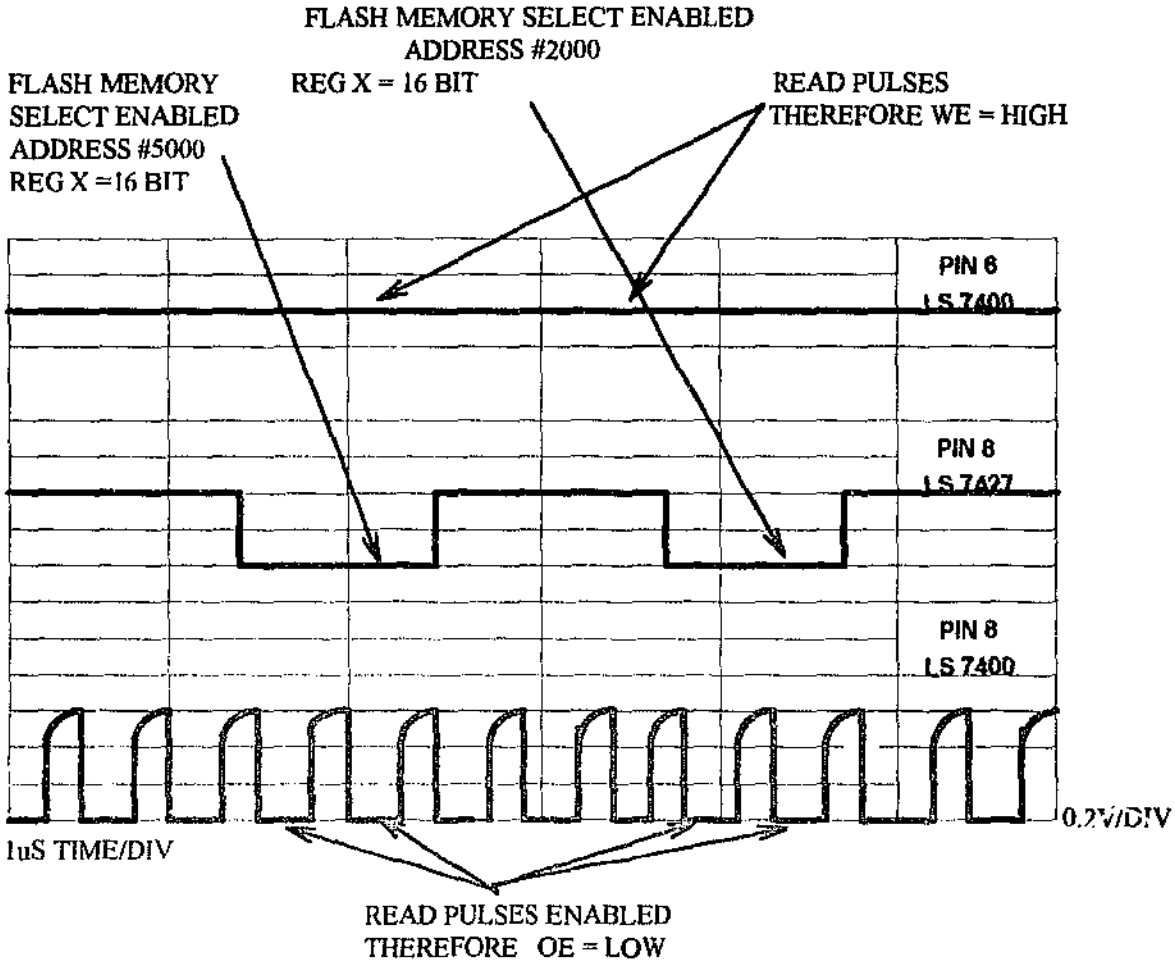
Program: : Testing of LS7400, LS7402, LS7427: Testing store instructions in range of FMS.

MEMORY LOCATION	INSTRUCTION	COMMENTS
0100	stx 5000	;writes value to Ports B & C inside of range of Flash Memory Select.
0103	stx 2001	;writes value to Ports B & C inside the range of Flash Memory Select.
0106	nop	;no operation, used to help distinguish signals
0107	nop	
0108	nop	
0109	bra 100	;branches back to 100, Infinite loop
010B	wai	;end of program.



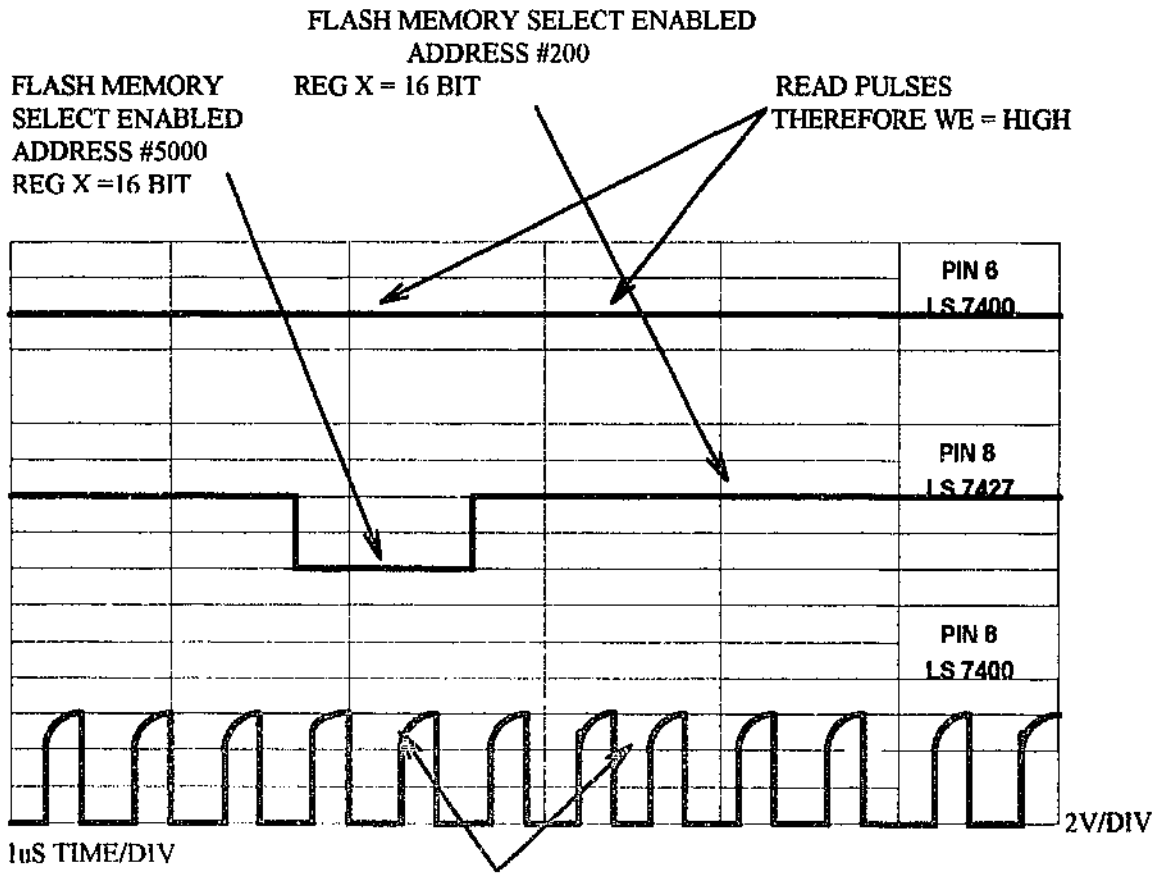
Program: Testing of LS7400, LS7402, LS7427: Testing 16 bit store instructions in range of FMS.

MEMORY LOCATION	INSTRUCTION	COMMENTS
0100	ldx 5000	;reads value from Ports B & C inside of range of Flash Memory Select.
0103	ldx 2000	;reads value from Ports B & C inside the range of Flash Memory Select.
0106	nop	;no operation, used to help distinguish signals
0107	nop	
0108	nop	
0109	bra 100	;branches back to 100, Infinite loop
010B	wai	;end of program.



Program: : Testing of LS7400, LS7402, LS7427: Testing store instructions in range of FMS.

MEMORY LOCATION	INSTRUCTION	COMMENTS
0100	ldx 5000	;reads value from Ports B & C inside of range of Flash Memory Select.
0103	ldx 200	;reads value from Ports B & C outside the range of Flash Memory Select.
0106	nop	;no operation, used to help distinguish signals
0107	nop	
0108	nop	
0109	bra 100	;branches back to 100, Infinite loop
010B	wai	;end of program.



CHAPTER 11

PIN CONNECTION BETWEEN THE EVB AND THE FLASH MEMORY BOARD.

After the placement of the integrated circuits on the MC68HC11 EVB, a connection between the chips and related pins needed to be achieved. The integrated circuits were connected using the Wire-Wrap process. After the Wire-Wrapping was completed a decision needed to be made on how the MC68HC11 is to be connected to the Flash Memory chips.

As there is not enough room on the MC68HC11 EVB to accommodated four Flash memory chips, an additional PCB holding the four Flash Memory chips was used. The board is a standard Wire-Wrap PCB with a 50 pin connector connected at the end of the board. The 50 pin connector on the Flash Memory board (which it will be referred to from now on) will be connected to the four Flash Memory chips via the Wire-Wrap method.

A connection now must be made between the MC68HC11 EVB and the new Flash Memory board. To achieve communication between the two boards a 50 pin Wire-Wrap connector was strategically placed at the end of the EVB. This 50 pin connector now enabled the MC68HC11 to be connected to the Flash Memory board. The communication media was via a 50 pin parallel ribbon cable. The cable was connected to the 50 pin connector on the MC68HC11 EVB and further attached to the 50 pin connector of the Flash Memory board. Both boards can now were able to communicate.

However, before any communication could be under taken, various signal lines must be associated to the various pins on the MC68HC11 EVB 50 pin connector. This pin and signal association must then correspond to the 50 pin connector on the Flash

Memory board which in turn must correspond to the relative pins on the Flash Memory chips.

A list of the various pin numbers of the associated chip to its corresponding pin number of the 50 pin connector is listed below. The list is further enhanced by correlating the pin number of the 50 pin Flash Memory board connector to its corresponding Flash Memory chip pin number.

FLASH MEMORY CONNECTOR: PIN LAYOUT USING THE PCB SUPPLIED BY E.C.U

WIRE WRAP SIDE

CHIP/PIN No	MC68HC11 Pin Connector	Flash Memory Board Flash Pin Connector No
Pre-Latch 74LS373	No	
3	3	13
4	5	14
7	7	15
8	9	17
13	11	18
14	13	19
17	15	20
18	17	21
Post-Latch 74LS373		
2	19	12
5	21	11
6	23	10
9	25	9
12	27	8
15	29	7
16	31	6
19	33	5
Port B A8-A14		
42	35	27
41	37	26
40	39	23
39	41	25
38	43	4
37	45	28
36	47	29

Flash Memory Connector: Pin layout using the PCB supplied By E.C.U

WIRE WRAP SIDE

Block Select		
PD3	8	2
PD2	10	3
74LS32		
3	40	Chip 1 pin 22
6	44	Chip 2 pin 22
11	46	Chip 3 pin 22
8	48	Chip 4 pin 22
74LS7400		
6	20 WE'	To all Chips 31
8	24 OE'	To all Chips 24
GND	60/59	16
VCC	1/2	1/2

On completion of the connection of the of the MC68HC11 EVB to the Flash Memory board via the 50 pin ribbon cable, testing of the two boards to ensure that there is communication between them needs to be under taken.

One procedure to test that there is communication between the MC68HC11 and the Flash Memory board is to test that the Output Enable, Write Enable, Block Select and the Chip Select pulses from the relative circuits reach the appropriate pins on the Flash Memory board. The test procedure to do this is described in the next chapter.

CHAPTER 12

MC68HC11 TO FLASH MEMORY P.C.B. TEST PROCEDURE:

The purpose of this test procedure is to check that the Output Enable, Write Enable, Block Select and Chip Select pulses from the relative integrated circuits reach the appropriate Flash Memory Chip pins on the Flash Memory board:

While the purpose of this test is to check that pin 2, 3, 22, 24, 31 on each Flash Memory chip has the right signal present at its pin, the test procedure also ensures that

- the LS7400, LS7402, LS7427, LS138 and the LS32 are functioning properly,
- the Cable connector is in working order,
- the PCB tracks are not obstructed,
- no other Flash Memory chip besides the chip designated receives a LOW for CS' (pin 22) , OE'(pin 24) and WE'(pin 31),
- the Wire-Wrapping on both the MC68HC11 board and the Flash Memory board is correct.

The following program was used to verify that the relevant signals were being reached by the chips.

Program: TO TEST THAT THE APPROPRIATE SIGNAL GOES TO THE CORRECT
CHIP PIN ON THE FLASH MEMORY BOARD, i.e.

- CHIP SELECT PULSE IS LOW,
- APPROPRIATE BLOCK OF THE FLASH MEMORY IS SELECTED,
- THE WRITE ENABLE SIGNAL IS LOW AND
- THE OUTPUT ENABLE SIGNAL IS HIGH

Note:

As the ranges above and below the Flash Memory Select have already been tested and proven to work, a test using memory locations within the range of the Flash Memory Select will be used in this program. If memory ranges between 2000 and 9FFF were not used no signals would be sent to the Flash Memory Board, thus, this exercise in testing would not work.

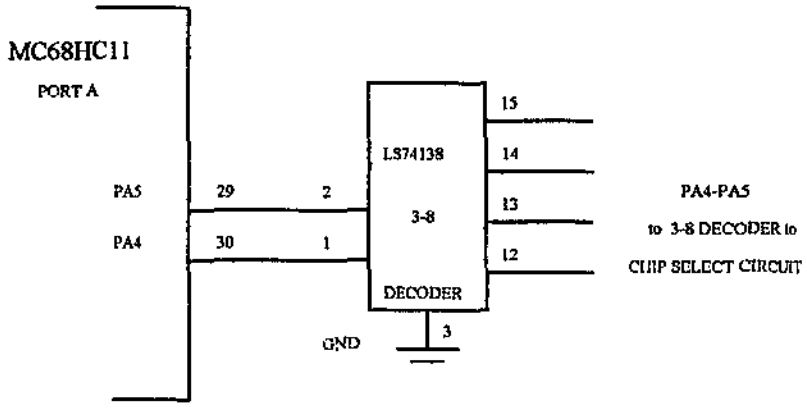
MEMORY LOCATION	INSTRUCTION	COMMENTS
0100	ldaa #0C	; sets Port D (pin 2, pin 3) pins to output control pins.
0102	staa 1009	; Port D control location.
0105	ldaa 20	; memory location 20 used to select which Flash Memory Chip.
0107	staa 1000	; Port A location
010A	ldab 21	; memory location 21 used to select which block is to be used within Flash Memory Chip.
010C	stab 1008	; Port D location
010F	nop	; no operation
0110	ldaa #5A	; data
0112	staa 3000	; Flash Memory location
0113	nop	; no operation
0104	nop	; no operation
0105	nop	; no operation
0106	nop	; no operation
0107	nop	; no operation
0108	bra 112	; branches to memory location #0112 for loop
0109	wai	;end of program.

TEST RESULTS

The following key is an interpretation to the results listed in the table. The term pulse is used to differentiate between a continuous LOW signal and a continuous HIGH signal. The Pulse is in conjunction with clock signal.

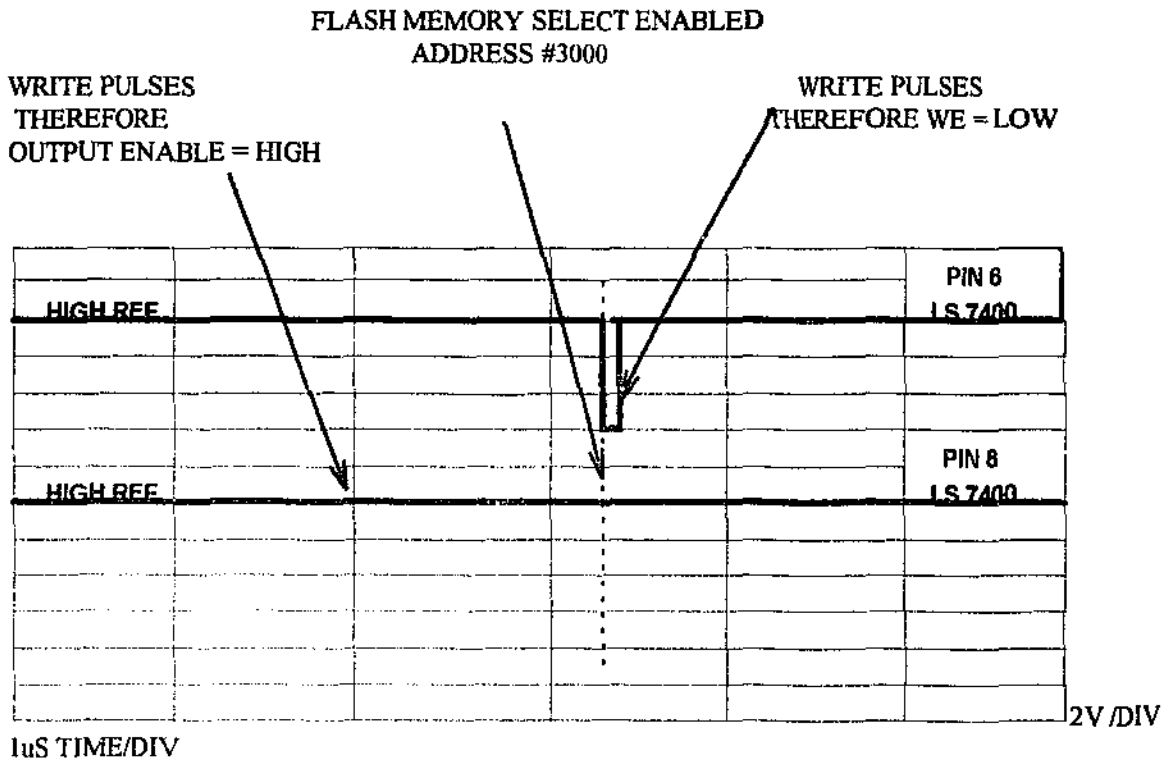
Key to test results:

H = High signal
L = Low signal
P = Pulse (in conjunction with clock signal)

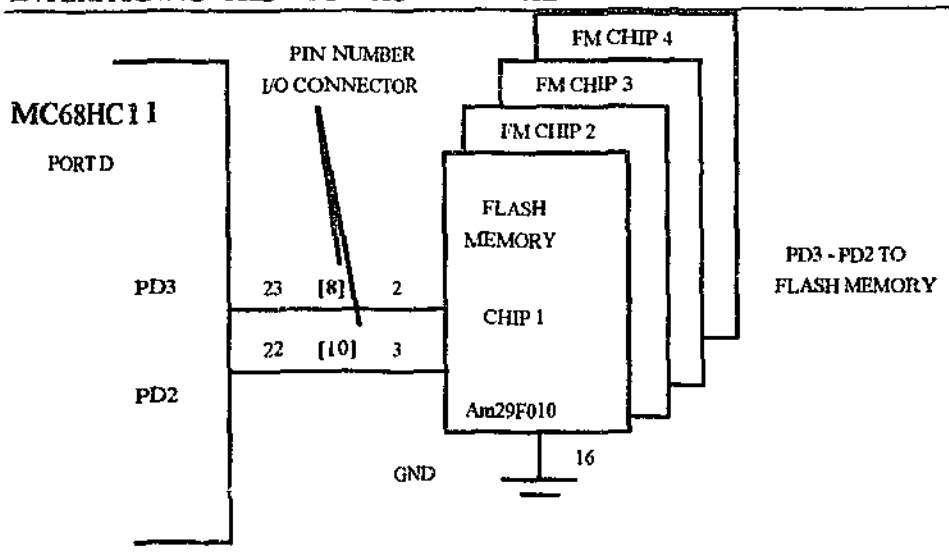


Memory location (data) 20 21	Chip LS 7427 pin 8 FM'	Chip LS 7400 pin 6 WE'	I/O CONNECTOR Pin-- nos 29-PA5	30-PA4	22-PD2	23-PD3
00 00	L	L	L	L	L	L
10 00	L	L	L	H	L	L
20 00	L	L	H	L	L	L
30 00	L	L	H	H	L	L

Results of Test procedure: Values of pins corresponding to FM', WE', PA5, PA4, PD3 and PD2.

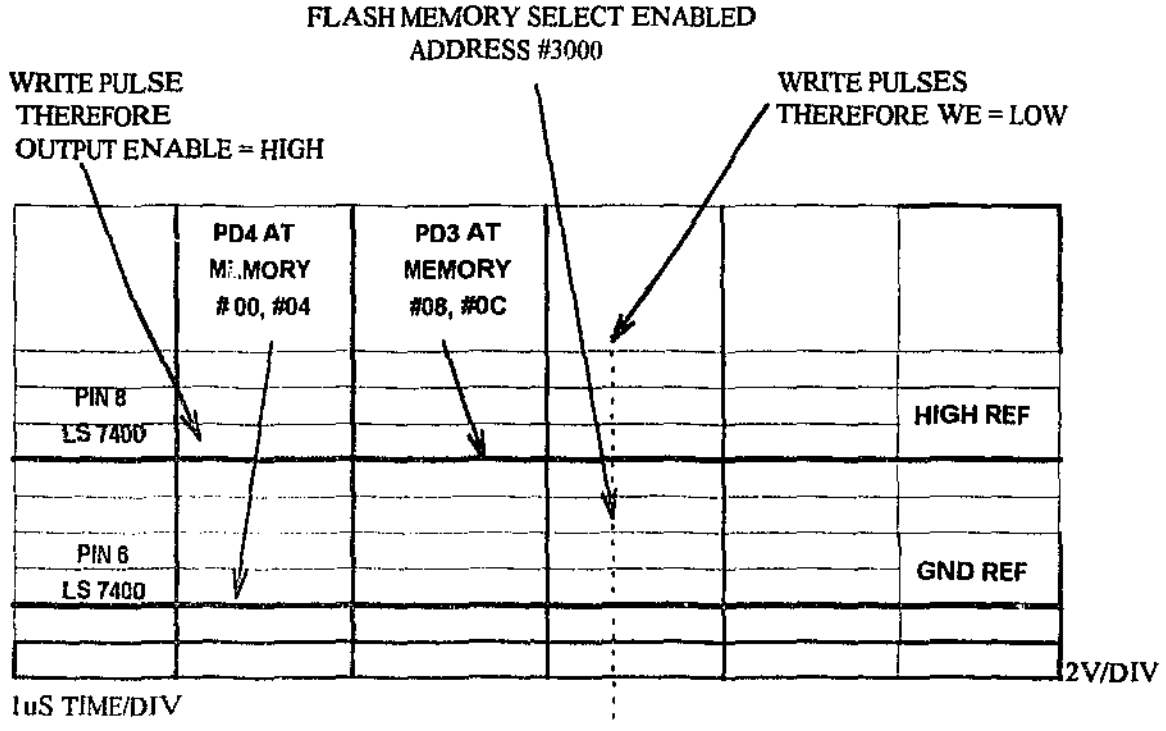


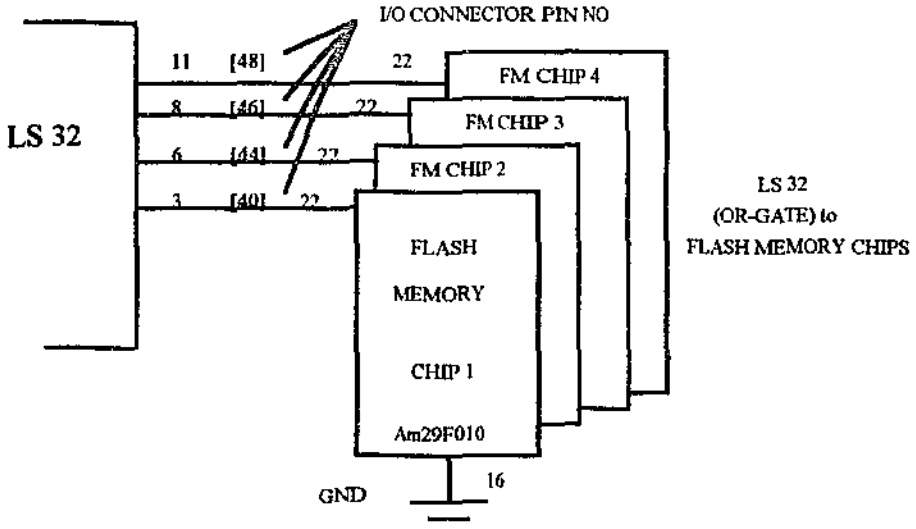
Test results of LS7400 Write and Output enable pulses.



Pins corresponding to FM', WE' PD3 and PD2 are tested are results recorded.

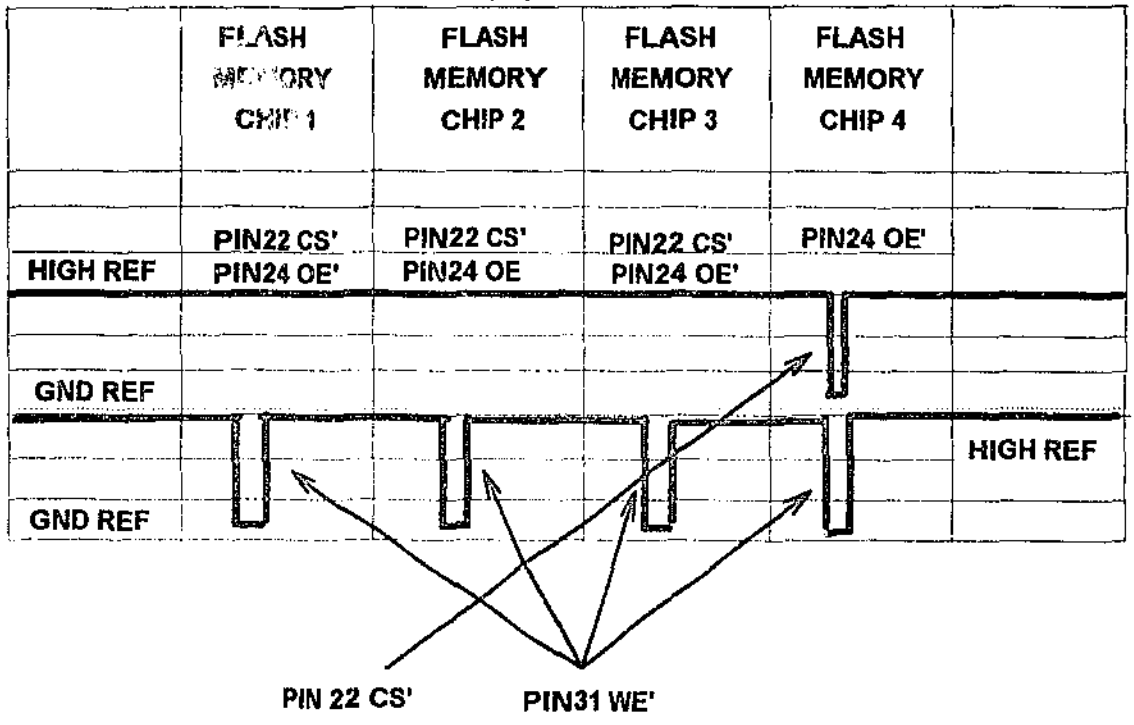
Memory location (data) 20 21	Chip LS 7427 pin 8 FM'	Chip LS 7400 pin 6 WE'	I/O CONNECTOR Pin-- nos 22-PD2 23-PD3
00 00	L	L	L L
10 04	L	L	L H
20 08	L	L	H L
30 0C	L	L	H H



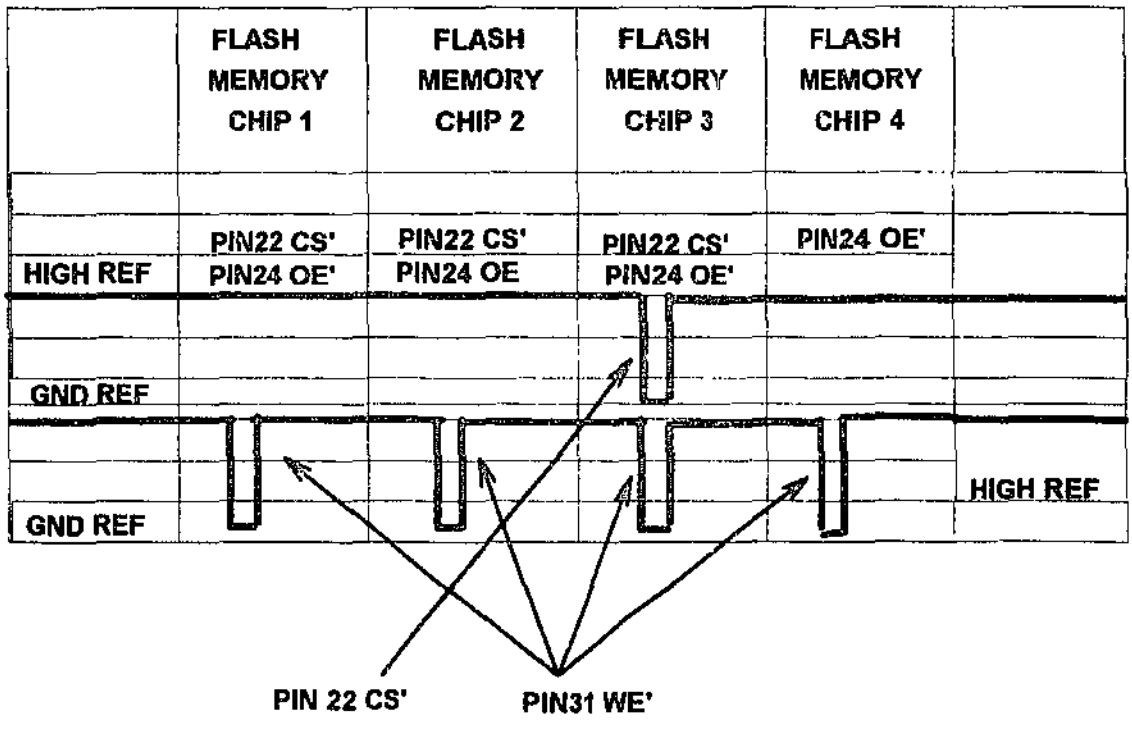


Memory location (data) 20 21	Flash Memory Chip 1 (22) (24) (31) CS' OE' WE'	Flash Memory Chip 2 (22) (24) (31) CS' OE' WE'	Flash Memory Chip 3 (22) (24) (31) CS' OE' WE'	Flash Memory Chip 4 (22) (24) (31) CS' OE' WE'
00 00	H H P	H H P	H H P	P H P
10 00	H H P	H H P	P H P	H H P
20 00	H H P	P H P	H H P	H H P
30 00	P H P	H H P	H H P	H H P

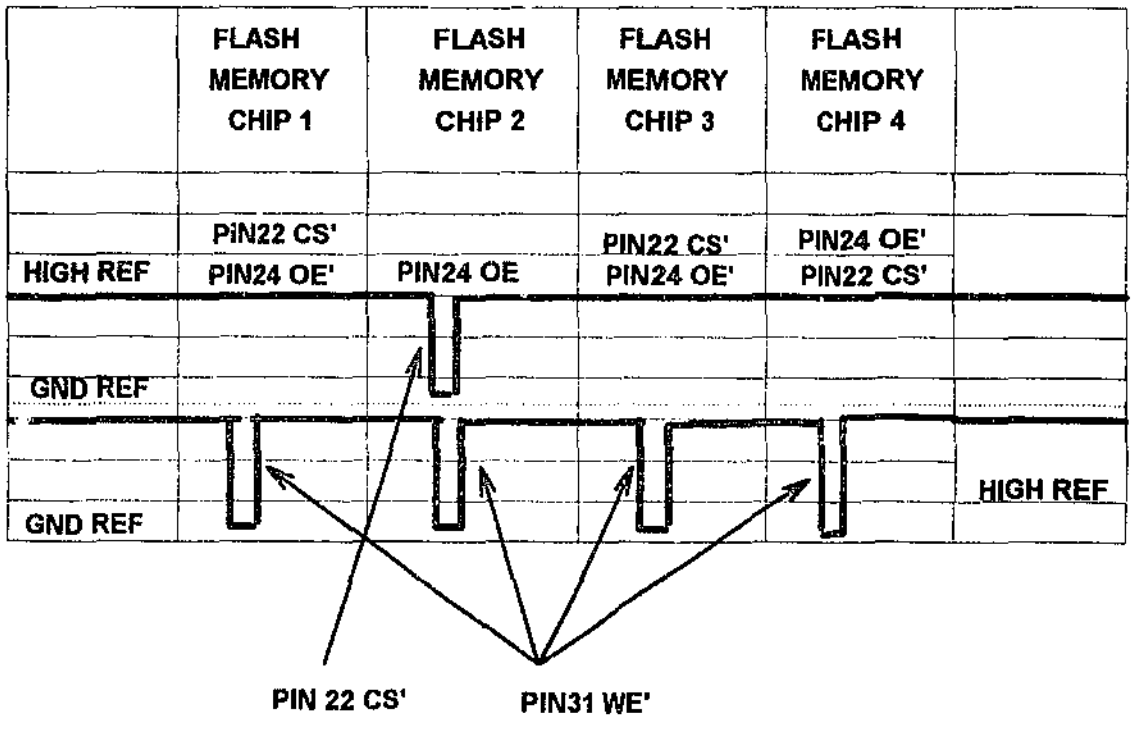
Pin number of AmF1010 in brackets (**)



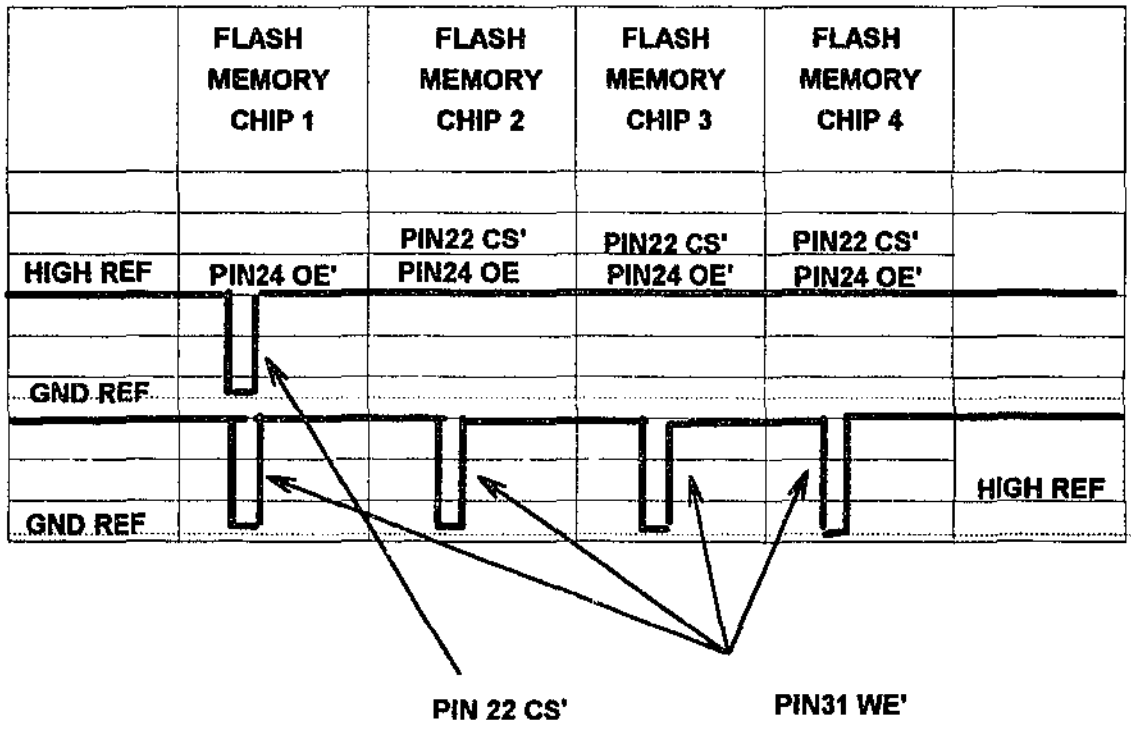
For memory location #00



For memory location #10



For memory location #20



For memory location #30

Program: TO TEST THAT THE OUTPUT ENABLE SIGNAL GOES TO THE CORRECT PIN ON THE FLASH MEMORY CHIP, i.e.

- CHIP SELECT IS LOW FOR THE APPROPRIATE CHIP,
- APPROPRIATE BLOCK OF THE FLASH MEMORY IS SELECTED,
- THE WRITE ENABLE SIGNAL IS HIGH AND
- THE OUTPUT ENABLE SIGNAL IS PULSED LOW

As the ranges above, below and inclusive of the Flash Memory Select have already been tested and proven to work, a test using memory locations within the range of the Flash Memory Select will be used in this program. Note: memory ranges between 2000 and 9FFF must be used in order to receive the appropriate signal at the relevant Flash Memory chip.

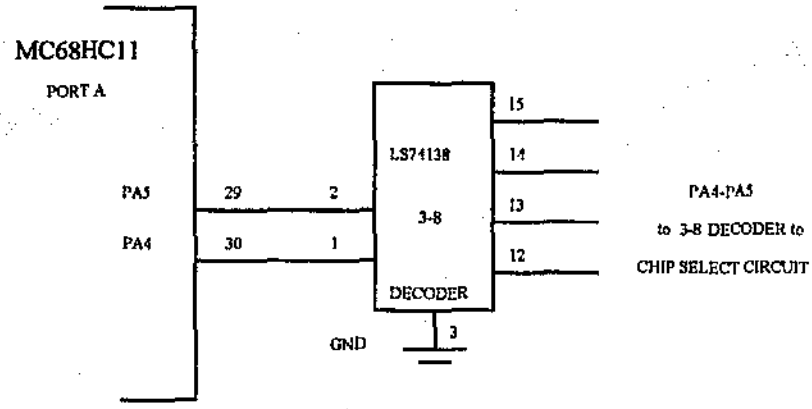
MEMORY LOCATION	INSTRUCTION	COMMENTS
0100	ldaa #0C	; sets Port D (pin 2, pin 3) pins to output control pins.
0102	staa 1009	; Port D control location.
0105	ldaa 20	; memory location 20 used to select which Flash Memory Chip.
0107	staa 1000	; Port A location
010A	ldab 21	; memory location 21 used to select which block is to be used within Flash Memory Chip.
010C	stab 1008	; Port D location
010F	nop	; no operation
0110	ldaa #5A	; data
0112	ldaa 3000	; Flash Memory location
0113	nop	; no operation
0104	nop	; no operation
0105	nop	; no operation
0106	nop	; no operation
0107	nop	; no operation
0108	bra 112	; branches to memory location #0112 for loop
0109	wai	;end of program.

TEST RESULTS

The following key is an interpretation to the results listed in the table. The term pulse is used to differentiate between a continuous LOW signal and a continuous HIGH signal. The Pulse is in conjunction with clock signal.

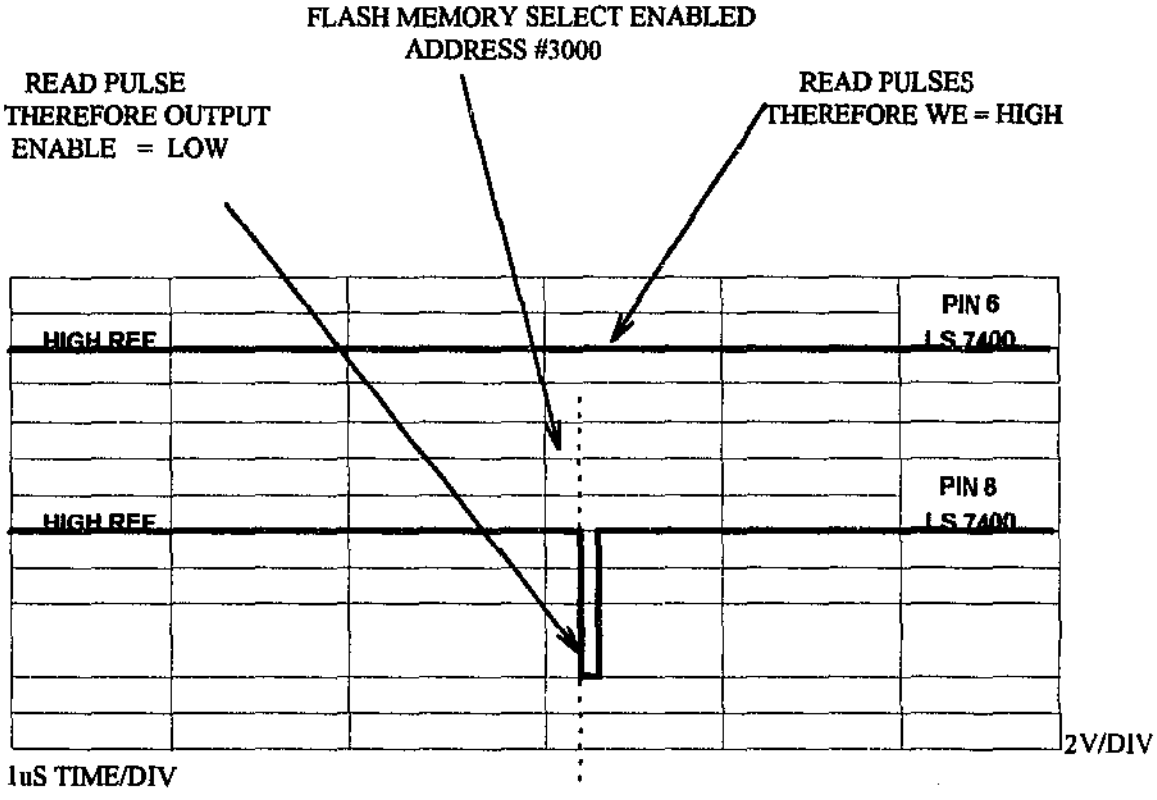
Key to test results:

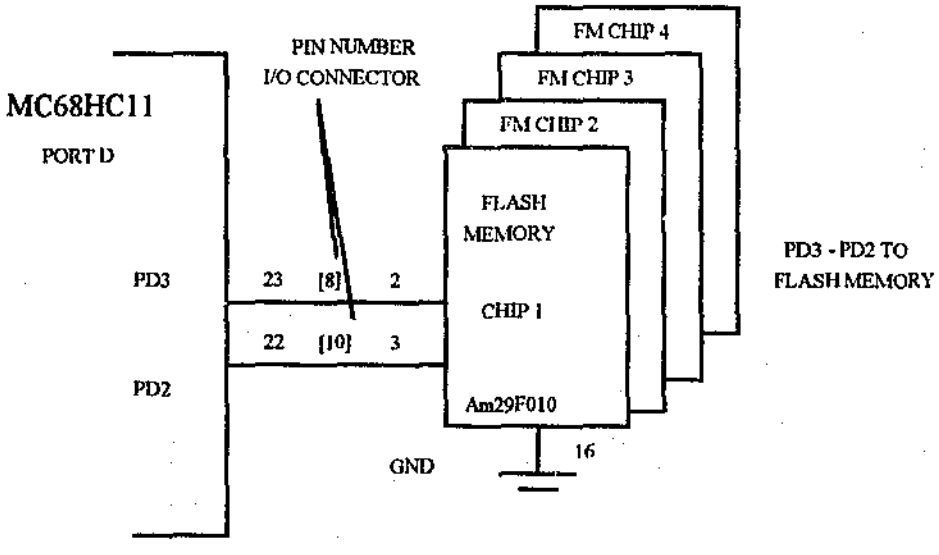
H = High signal
L = Low signal
P = Pulse (in conjunction with clock signal)



Test results of pins corresponding to FM', WE', PA5, PA4, PD3 and PD2.

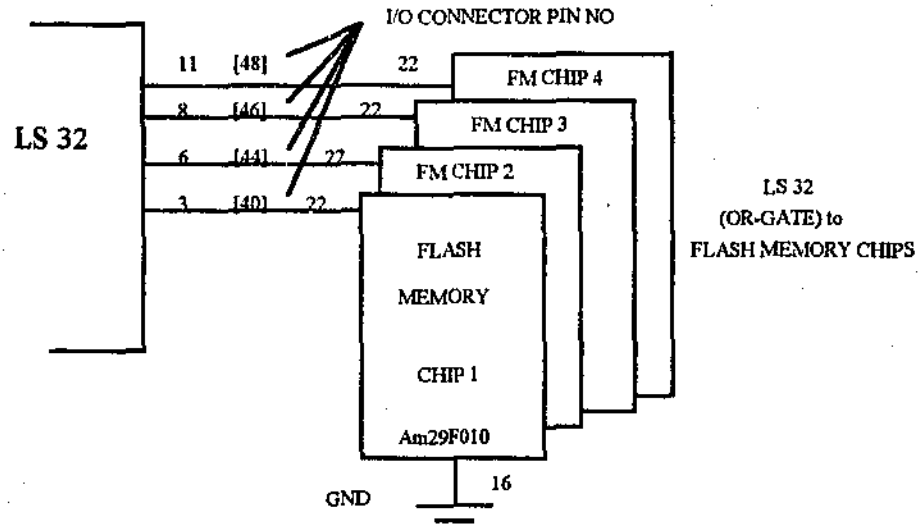
Memory location (data)		Chip LS 7427 pin 8	Chip LS 7400 pin 6	Chip LS 7400 pin 8	I/O CONNECTOR Pin-- nos			
20	21	FM'	WE'	OE'	29-PA5	30-PA4	22-PD2	23-PD3
00	00	L	H	P	L	L	L	L
10	00	L	H	P	L	H	L	L
20	00	L	H	P	H	L	L	L
30	00	L	H	P	H	H	L	L





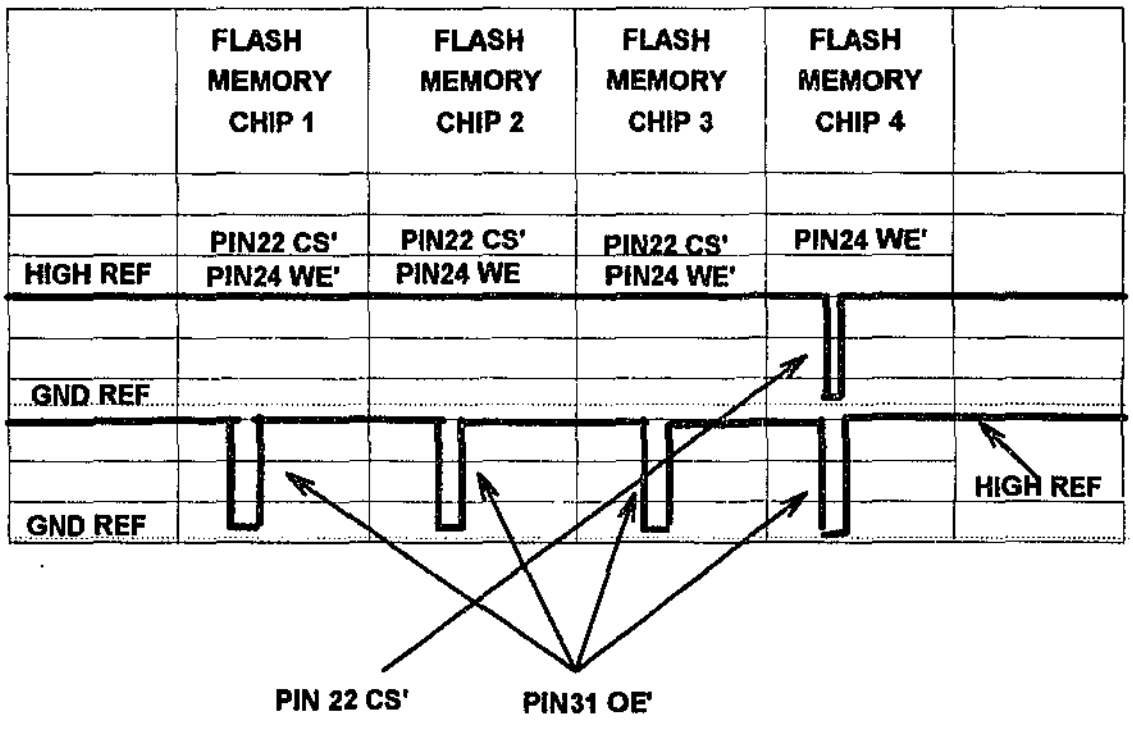
Here pins corresponding to FM', WE', PD3 and PD2 are tested are results recorded.

Memory location (data) 20 21	Chip LS 7427 pin 8 FM'	Chip LS 7400 pin 8 OE'	I/O CONNECTOR Pin-- nos 22-PD2 23-PD3
00 00	L	P	L L
10 04	L	P	L H
20 08	L	P	H L
30 0C	L	P	H H

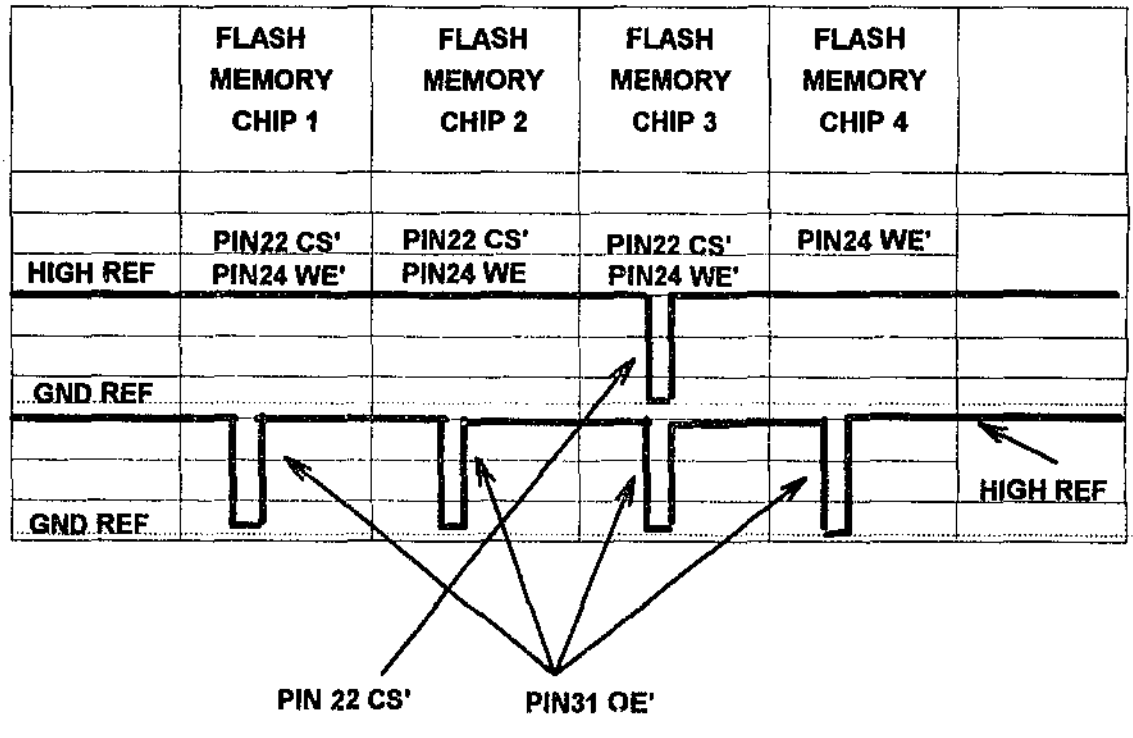


Memory location (data) 20 21	Flash Memory Chip 1 (22) (24) (31) CS' OE' WE'	Flash Memory Chip 2 (22) (24) (31) CS' OE' WE'	Flash Memory Chip 3 (22) (24) (31) CS' OE' WE'	Flash Memory Chip 4 (22) (24) (31) CS' OE' WE'
00 00	H P H	H P H	H P H	P P H
10 00	H P H	H P H	P P H	H P H
20 00	H P H	P P H	H P H	H P H
30 00	P P H	H P H	H P H	H P H

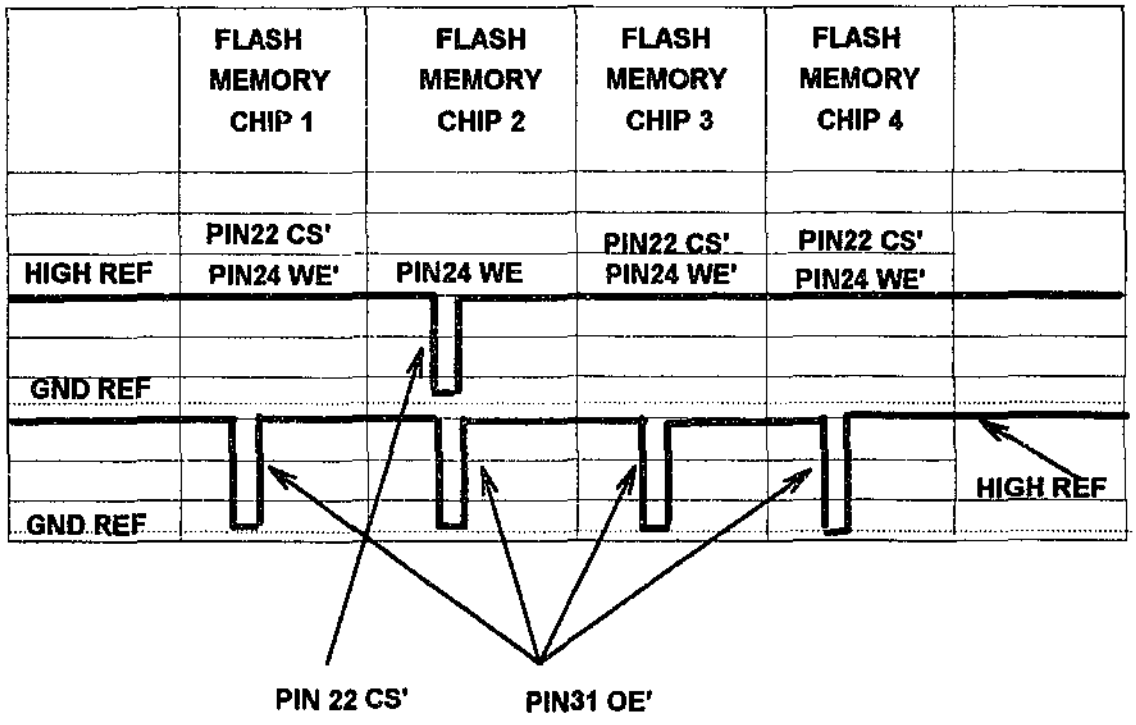
Pin number of AmF1010 in brackets (**)



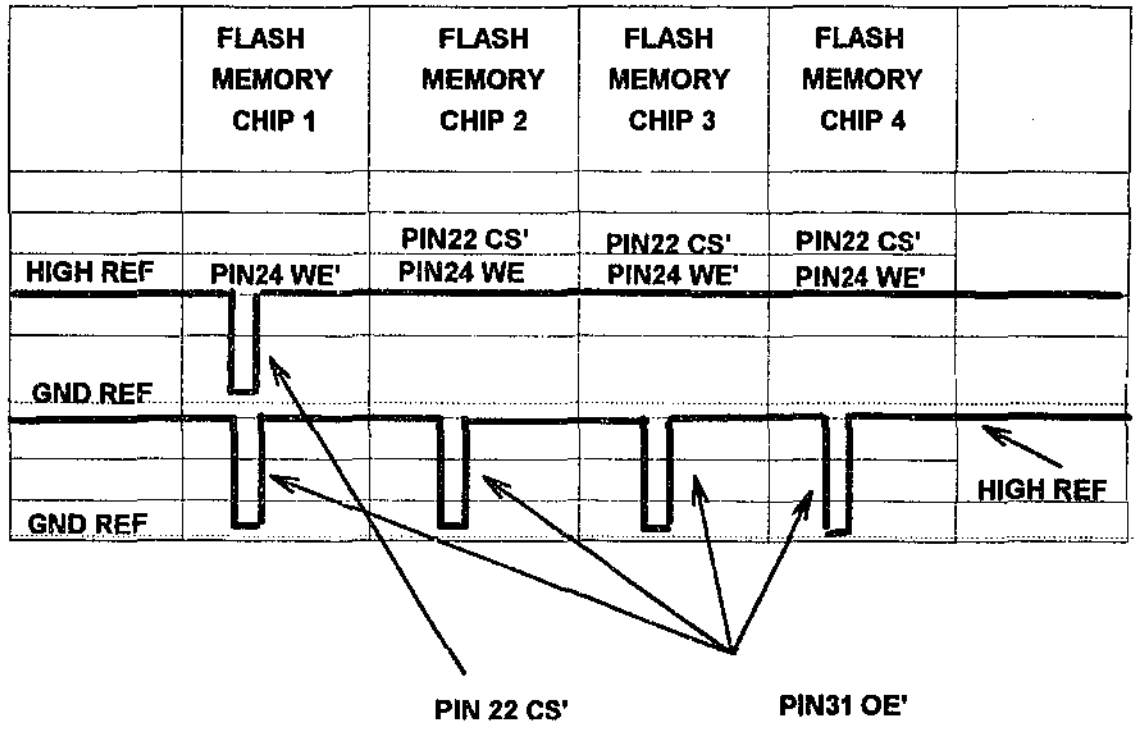
Test results CS', OE', and WE' for memory location #00



Test results CS', OE', and WE' for memory location #10



Test results CS', OE', and WE' for memory location #20



Test results CS', OE', and WE' for memory location #30

CHAPTER 12

SCSI OR IDE?

INTRODUCTION

The accelerated and perpetual development of more sophisticated software has now caused a need for more storage capacity. Today, the ability to do serious work without a hard disk is limited. In essence, most programs that do not require a hard disk are of little value. If by chance these programs are of any caliber, their only fate is to go through a long tedious procedure to achieve their objectives. Resulting from this swelling in the sophistication of software applications a chain reaction evolved causing an increased demand in hard drives with large storage capabilities. Real-time data acquisitions also augmented the obligation for hard disks to increase their storage capacities. After the need for high performance hard disks was established, the disk controller came into the spotlight as both are closely related.

A disk controller is a circuit board connected to a PC or Microcontroller's bus that receives signals from the CPU and translates them into a format that can be understood by the disk hardware (Doty D.B,1993). The controller is responsible for the moving of blocks between the disk and the CPU. Disk controllers also specify the encoding format and the interface standard they use. An interface standard specifies what type of signals a hard disk controller will send and receive and what type of cabling and connectors the controller will use to connect to the disk.

Currently there are four different hard-disk interface standards in use: ST506, ESDI, IDE (integrated drive electronics) and SCSI (small computer system interface). The earliest of these standards is the ST506. Found in the earlier systems, the ST506 was a product produced by the now deceased Shugart Company (Doty D.B,1993). ESDI (Enhanced Small Device Interface) an updated type version of the ST506, supported

larger dives and faster data transfer than the ST506 but was edged out by the two latest standards, IDE and SCSI.

IDE originated by Compaq in 1984 became an interface standard (Doty D.B,1993). IDE also came known as AT drives because they were first developed for use on the 286 AT (Pilgrim A, 1993l). In 1986 SCSI-1(pronounced "scuzzy") became an official ANSI standard (http://www.cis.ohi... faq/part1/faq.html, 1996).

Due to the advent of CD-ROM, scanners ,tape drives and other peripheral devices, IDE and SCSI spawned from standardised hard disk interfaces to standard peripheral interfaces. Further born from this standardisation of the two interfaces came a great industry debate of SCSI versus IDE. A decision now exists as to which controller to use to interface the Microcontroller-based Flash Memory Secondary Storage Device, SCSI or IDE. The following discussion investigates the concrete differences between.

IDE AND SCSI COMPARED

Described as the "nuttiest business in the world" (PC Magazine July 1995) the competition between IDE and SCSI in its bid to make it to the desktop was razor sharp and cut-throat. Through this competition advancements in both IDE and SCSI the seeded foundation for new standards and specifications propelled into the market with impelling force. Advances in IDE now give Enhanced IDE (EIDE), E-IDE-95, Fast-ATA, ATA-2, ATA-3 and ATAPI. Advances in SCSI now donate SCSI-2,W SCSI-2, F SCSI, SCSI-3 subset and several others up to double speed SCSI. For the purpose of the this project this discussion will be limited to EIDE and SCSI-1 as most of the advancements past these two have not been fully benched marked or variation of parameters is unclear as information differs from article to article.

IDE AND EIDE

After Compaq approached Imprimis(which is now part of Seagate) in 1985 and Imprimis succeeded in integrating the Western Digital circuitry onto the Wren drive controller board, IDE has become the most popular disk interface in the PC environment (BYTE MARCH 1991). It is estimated that 95 percent of all PC disks are standardised on IDE (Computer Shopper, May 1994). The main reason validating this is that IDE offers easier technology to implement and as such becomes the least expensive solution.

At the time of IDE's introduction no-one perceived the enormity in growth that technology would under go. As a result, no consideration was given to the significance of the performance of IDE in the future. Also, at the time after its introduction no-one considered that these imitations in IDE's capabilities would soon become a significant factor co-hesive to the rapid growth of technology. Because of this sudden surge in technology new trends in new devices were brought about causing a re-assessment of IDE's limitations. Trends such as the ones listed below have now drawn severe limitations using IDE;

- *Multitasking* - IDE is single threaded by nature (Computer Reseller News, 1994). The current command must complete before additional commands can start. With most IDE adapters the processor must be involved in reading/writing the data from/to memory. As IDE has an inability to allow more than one transfer of data at a time it significantly places itself at a distance from multitasking operating systems. The request of data from the disk by the interface may take between 15 - 20 ms, which can be considered a long time in processor terms. This inability to request another piece of data while waiting for the first to be returned means data is handled less efficiently, resulting in reduced

- overall disk performance. The current generation of operating systems, such as OS/2, MS-Windows 95, and MS Windows NT are all multi tasking environments which are not well suited to IDE interfaces.

- *Number of devices supported* -Another drawback is that only two magnetic hard disks can be attached (Windows Sources, May 1994). In a single drive single-tasking system IDE will probably be slightly faster and is definitely less expensive. Cable length a maximum of 18 inches or 45cm (Computer Shopper, May 1994),

- *Multimedia* - because of their sound, large video, and graphics files, multi media systems require very large hard disks. Under MS-DOS, IDE hard disk drives are limited to 528 MB in size (Computer Shopper, May 1994),

- *ROM* - demand for low expense CD-ROM solutions continues to grow at a rapid rate. IDE has the ability to only support hard disk drives and not CD-ROM drives (Windows Sources, May 1994). At present CD-ROM uses either a SCSI interface or a proprietary interface supplied with an add-on card delivered with the drive.

- *Local Bus Technology* -data transfer rates using the standard IDE is 2-3 MB/sec which is relatively low to that of SCSI's 10MB/sec (Computer Shopper, August 1994). The combination of local bus technology, e.g., Peripheral Component Interface (PCI) and VL-Bus, with fast pentium and Intel Dx4 processors have exposed the hard disk subsystem as a potentially serious bottleneck.

Due to these new trends in technology IDE fell short in some of its capabilities when compared to SCSI. Enhanced IDE (EIDE) or AT-2 was an interim solution developed to provide improved disk performance. EIDE proposed by Western Digital (http://www.buds.co.uk/pc-tech/ide-scsi) became a new standard designed to overcome the limitations of IDE. It further more increased its capabilities relation to SCSI, but did not incur the inherent cost factor.

THE BENEFITS OF ENHANCED IDE

From the ever increasing technological growth great demands were put on IDE to keep ahead of its competitor, SCSI. With this competition came the growth of ENHANCED-IDE (EIDE), which is now the new standard proposed by Western Digital. However. while designed to overcome the short falls of IDE Western Digital, Quantum, Maxton and several other companies also endeavored to retain its competitive edge without incurring the more expensive cost of SCSI based solutions (Computer Shopper, Sept. 1994).

In comparison to IDE the benefits of EIDE are listed below:

- *Larger drives* - EIDE supports drives of up to 8.4GB in size(Computer Shopper, Sept. 1994). Previous IDE could not support disks larger than 528 Mbytes.

- *Multiple devices* - EIDE will now allow up to four devices to be connected at any one time, instead of one, as its predecessor IDE(Windows Sources May 1994).

- *Support for other device types* - Significantly reducing the cost of CD-ROM accessories and simplifying installation for users, EIDE provides

- support for CD-ROM drives and tape backup systems. As a result, it is no longer necessary to supply a SCSI interface with a CD-ROM drive as was previously called for.

- *Faster data transfers* - EIDE interfaces and drives now support increased data transfer rates. By using Programmed Input/Output modes (PIO), the processor can itself control all transfers to and from the disk. PIO supports data transfer rates of up to 13.3MB per second(Computer Shopper, Sept. 1994), which is several times faster than standard IDE and faster than most SCSI interfaces.

- *Cost Factor* - EIDE brings significant advantages that can be implemented at a relatively low cost. At the time of writing the cost of a SCSI controller about $25 to manufactures relative to EDIE's $10 (Retail Research Western Australia). CD-ROM drives in particular are greatly affected by the price war. The cost savings from not having to supply an interface with the drive are significant.

- *Compatibility with previous IDE systems* - Already existing IDE drives can be attached to EIDE controllers with relative ease and still reap the fortunes of EIDE.

The capabilities and improvements from IDE to EIDE have been covered. For a fair judgment to be made upon which interface to use in the microcontroller-based Flash Memory Secondary Storage Device, SCSI now needs to be analysed for comparison.

BENEFITS OF SCSI

As mentioned previously IDE is the most popular disk interface in the PC environment. The main reason constituting this is that IDE is the more simpler out of the two technologies, therefore easier to implement and as such is less expensive. SCSI, on the other hand is harder to implement as it is more complex in technology and thus produces a higher cost factor. This factor of expense plays an important role in our choice of interface as the cost of any business or personal purchase effects our economic climate.

Described as competitive "treadmill they can't get off" (PC Week, July 1994), competition plays a vital role in the survival of any manufacturer. Staying ahead of a competitor in terms of cost is a key factor in keeping a business prosperous. However in most cases cost is usually a trade off for quality and in the electronics industry a trade off for performance. Because IDE is simpler in technology and easier to implement than SCSI, limitations in its performance are eminent when the two are compared. The integrity of this prediction of shortfalls inlimitations in being a trade off for expense is in fact a reality, IDE has greater limitations when compared with SCSI.

SCSI supports very large disks, permits multiple simultaneous bus transfers and allows a number of different device types, such as scanners, CD-ROM drives, and printers, to be connected to it. Basically it has none of the limitations of IDE and when comparing it to EIDE represents an excellent low-cost solution, there are still several important areas where SCSI retains a significant advantage.

- *Multitasking* - SCSI allows multiple I/O requests and allows them to be completed in any order. For instance, if process 'X' needs to read a block. The request is sent to the drive, the disk head starts to move, and process 'X' blocks are waiting for it. Then, process 'Y' is allowed to run; it also

- reads a block from the disk. Process Y's block may be sitting in a RAM cache on the SCSI controller, or on the drive itself. Or the block may be closer to the head than process X's block, or on a different drive on the same cable. SCSI allows process Y's request to be completed ahead of process X's, which means that process 'Y' can run sooner, so that the CPU (the systems most expensive chip) spends less time idle. Under IDE, the process Y's request cannot be sent to the drive until the process X request is complete. These SCSI capabilities are very valuable in a true multi-tasking environment, especially important in a busy file server, and useless under DOS (which cannot take advantage of them). Multitasking- Under DOS (and DOS/win3.1), there is very little useful work the host can do while waiting for a disk operation to complete. So handing off some work from a 66 MHz 486 to, say, an 8 Mhz Z80 (on the controller) does result in a performance loss. Under every other OS worth discussing (UNIX, Netware, NT, OS/2, Win95 etc) the processor can proceed to do something else while the access is in progress. This increases the performance by the CPU. In such systems (due to virtual memory), a 64K byte 'contiguous' read requested by a process may be spread to 16 separate physical pages. A good SCSI controller, given a single request, can perform this 'scatter/gather' operation autonomously. For high-end servers and power users running operating systems such as UNIX, Windows NT, or OS/2, SCSI is the only choice.

- *Disk performance*- Where speed lies as an important factor (e.g. high-end desktop machines and servers), SCSI retains a performance edge. Enhancements such as the SCSI-2 Fast/Wide interface deliver data transfer rates of up to 20MB per second (Computer Shopper, May 1994),

- *Flexibility*- SCSI supports a wider variety of devices, and allows these devices to be physically external. EIDE allows a maximum cable length of 46cm - insufficient to support external devices whereas SCSI cables can be up to 6 meters long (Computer Shopper, 1994). Therefore, systems needing to support large disk arrays, optical jukeboxes, or other similar devices, SCSI continues to be the only option.

- *Number of devices supported*- SCSI supports up to seven devices(Computer Shopper, Sept.), whereas Enhanced IDE can only support a maximum of four(Windows Sources May 1994). SCSI devices can be daisy-chained together, whereas all IDE devices must be directly connected to the controller.

However, the down side to SCSI is that SCSI is more complex to implement and thus creates a higher cost factor. In addition all SCSI devices must be intelligent devices with their own memory. This makes SCSI-based devices and systems, as well as SCSI cables and connectors, more expensive to manufacture than their IDE equivalents.

Which disk interface is recommended to incorporate The Micro-controlled Flash Memory Secondary storage Device, SCSI or Enhanced IDE?

CONCLUSION

Having discussed the advantages and disadvantages of both SCSI and IDE the ultimate decision of which controller to use now needs to be made, but does this choice just lie with one or the other ?

The ingenious founders of physics and maths always drew some conclusion that inferred only 'one' right possible explanation or outcome. Since then the result of some investigation, whether it be mathematics, physics or some other science usually draws the inherent 'one' answer explanation. Because this investigation is integrated to the science of engineering one would expect one and only one explanation. This general standard now leads us to make one and only one choice when choosing which controller to use, SCSI or IDE, in the Flash Memory Micro-controller Secondary Storage Device system.

This question of which controller to use however is to difficult to answer inferring one and only one solution. The answer would totally depend on how the system in use is employed, what operating system is installed, and if further I/O devices are likely to be added in the future.

For server systems that operate in a corporate environment the only solution would be to use SCSI. If adding more than four I/O devices in the future SCSI is the only choice.

The choice needs should be made on an individual basis and careful consideration should be given to all the issues that display the difference between SCSI and IDE. When using IDE problems are inevitable if a decision is made to add on another I/O device. If the choice is made to go with IDE, a question of whether the need to ever add a CD-ROM, CD-R, scanner, tape drive or need more than two hard disk drives in

the future needs to be answered . If SCSI is chosen could it have been a superfluous use of capital, as the PC will never be used for anything other than a word processor.

Another question to be answered is will the system ever be used for multitasking. Multitasking here though incorporates more than just actively running two or more programs at once. Consider the process when a window which uncovers parts of four other application windows is minimised. Each of the applications is sent a message telling it to update part of its window; under win95, they will all run concurrently to perform the update. If they need to access disk (usually because of virtual memory) the smoothness of the update can depend greatly on the disk system's ability to respond to multiple independent read-requests and finish them all as quickly as possible. When multi-tasking operating systems (like Win95, WinNT, Unix, OS/2 and Netware are used SCSI would be more advantageous). SCSI leads the way against IDE at this. ATA is faster under DOS; but SCSI provides advantages which are inaccessible to DOS.

The cost of intelligent, fast SCSI controllers and drives is likely to decrease as a result of exposure to these advantages of SCSI ahead of EIDE or IDE and therefore the cost of SCSI will be sure to fall.

Unlike the explanations of physics and mathematics there is no on distinct answer when choosing which controller to use, SCSI or IDE, in the Flash Memory Micro-controller Secondary Storage Device system. The decision has to be based on the application that the particular Flash Memory Micro-controller Secondary Storage Device system will be used for. Whether choosing SCSI, IDE or even EIDE, expense, updatability, multitasking and other options all have to be taken into account when making the decision.

CONCLUSION:

The objectives of this project was to design an interface from the MC68HC11 to four AMD Am29F010 Flash Memory chips. These objectives have now been achieved and proven to work through actual hardwiring of the chips to the MC68HC11 E.V.B. Because of this there is now and endless amount of applications that can use this new semiconductor extended memory. Applications such as Voice Recorder, Digital Video's, Voice/data pagers, Cellular phones, Mobile answering systems, PDAs, and Portable scanners are just a few that benefit from the new Flash Memory Secondary Storage Device System.

Note, that testing of recording of information to and from chip has been omitted from this report as if all signals tested are present at the respective inputs and control lines the flash memory will no doubt perform its assigned task.

Conclusions for each chapter objectives are set out below:

Chapter 1:
Mainly concerned with the History of Microcontrollers and their development to the present day, but, more importantly contained the Rationale for the whole project. It was set out, implemented tested and achieved.

Chapter 2:
An overview of the MC68HC11 for the purpose of revision to the reader and outlying important features that will be used in the project.

Chapter 3:
Introduced Flash Memory, its history, capacity, it architecture and its applications. It also illustrated why Flash Memory is of more benefit than other memory structures.

Chapter 4:

Reported the beginnings of the interface design. This section incorporates memory ranges that the MC68HC11 uses Flash Memory. Tested and achieved.

Chapter 5:

This chapter incorporated the design that chose which Flash Memory Chip is to be chosen out of the four. Design tested and achieved.

Chapter 6 :

Described the functions and commands of the ADM Am29F010 Flash Memory Chip

Chapter 7:

Design of how access to the address within the Flash Memory Chip was designed . Tested and achieved.

Chapter 8:

Description of READ/ WRITES made from the MC68HC11 to the Flash Memory Chip are explained in this section.

Chapter 9:

The important task of hardware construction using the Wire-Wrapping method is explained in detail in this chapter.

Chapter 10:

Design of the interconnection between the MC68HC11 board and the additional Flash Memory Chip Printed Circuit Board. Tested and achieved.

Chapter 11. Overall testing of the communication from the MC68HC11 to the four Flash Memory Chips.

Chapter 12 A detailed discussion involving using SCSI or IDE to interface as the new controller for this new secondary storage device.

LAST THOUGHT

Fifty years ago a computer occupying the size of a square centimeter was once thought of only in the minds of science fiction writers. However, this fiction is now a reality and the new Flash Memory Secondary Storage Device helps further this thought by making the computer even more powerful than it already is. This reality though is truly indebted not only to the writers who imagined it but to people who turned these imaginative thoughts into a reality.

BIBLIOGRAPHY

Arnold, J. 1996 Electronic Engineering Times June 17, n906 p79(1) COPYRIGHT CMP Publications Inc. 1996

Balch, K 1996 Roundtable: get it done in a flash Jan 1, v12 n1 p115 (4) COPYRIGHT CMP Publications Inc.

BYTE MARCH 1991

Computer Shopper, August 1994, v14 n8, pg 168, Ziff Davis Publishing 1994).

Computer Shopper, Sept. 1994, v14 n9, pg 482, Ziff Davis Publishing

Computer Shopper, Sept. 1994, v14 np, pg 62, Ziff Davis Publishing.

Computer Shopper, May 1994, v14 n5, pg 581, Costal Associates Publishing LP.

Computer Reseller News , June 20, 1994, n583, pg 90 Costal Associates Publishing

DataQuest ,1994 July pg. 42-50.

Doty D.B,1993. Power of Norton Utilities 7.0, pg 344, MIS press N.Y .

EDGE: Work-Group Computing Report April 29, 1996 v7 n311 p16(1)

EDGE: Work-Group Computing Report1996 Intel & Sharp introduce flash memory family on 0.4-micron; partnership promotes flash availability and uniform architecture, April 29,: v7 n311 p16 COPYRIGHT EDGE Publishing 1996

EDGE: 1996 Work-Group Computing Report April 29, v7 n311 p16(1) EDGE Publishing.

Horowitz,1989, The art of digitalelectronics , N.Y Cambridge University Press

http://www.buds.co.uk/pc-tech/ide-scsi

http://www.cis.ohi... faq/part1/faq.html, 1996.

IEEE, 1993 IEEE spectrum, October 1993, pg 48.

McCrindle, J. 1990 , Smart Cards. UK: IFS Publications/Springer-Verlag, pg. 11.

McCrindle, J. 1990, Smart Cards. UK: IFS Publications/Springer-Verlag, pg. 13 .

PC Magazine July 1995 v14 n13 p89

PC Week, July 18 1994, v11 n285, pg 37, Ziff Publishing Company 1994,

Pilgrim A , 1993 Build your own 486/486SX and save a bundle ,Windcrest /McGraw-Hill.

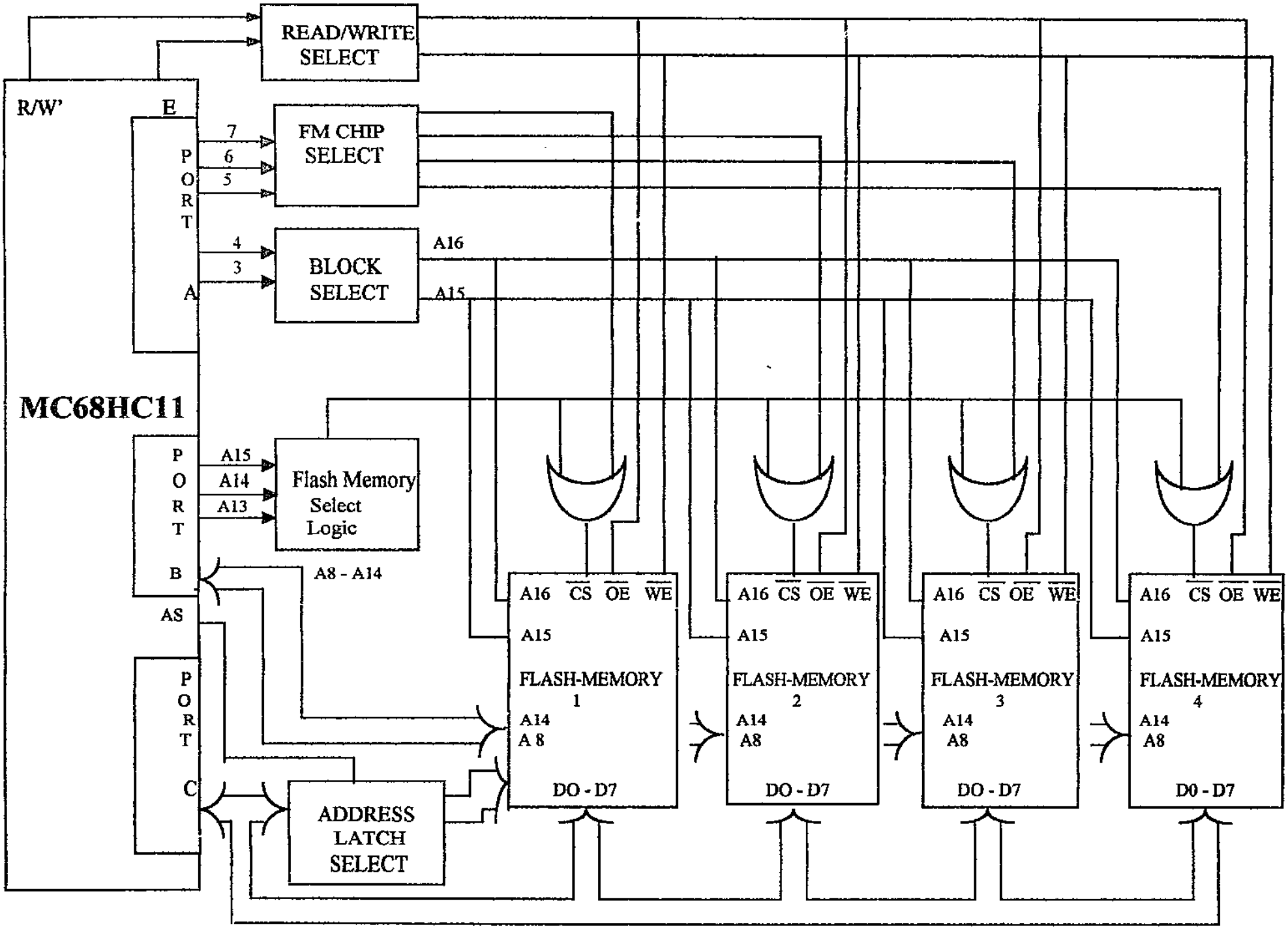
Olsen ,1977, Electronics made simple , G.B , Allen MC68HC11 Co

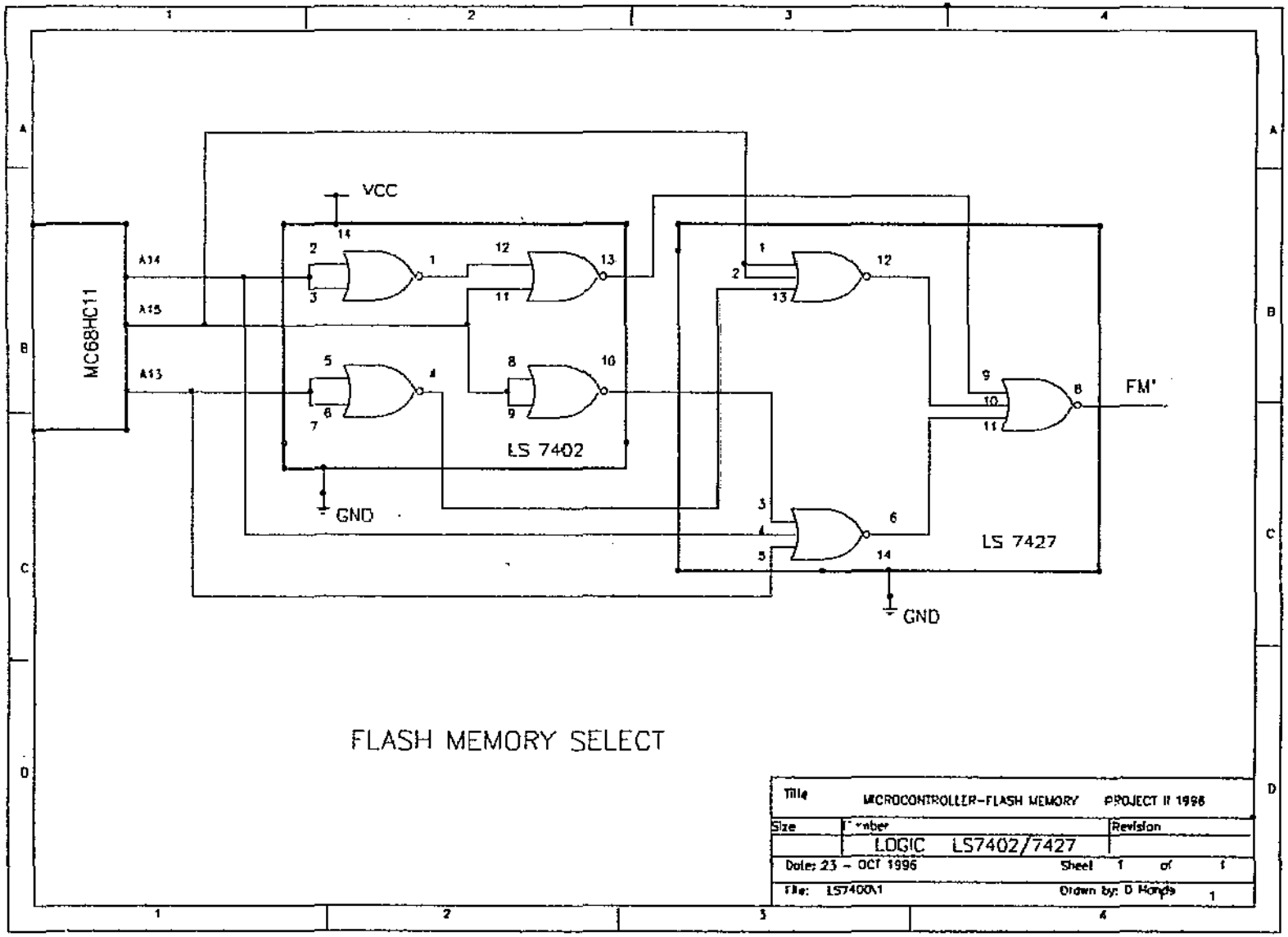
Spasov, P. 1992, Microcontroller Technology. N.Y: Prentice-Hall, pg 5.

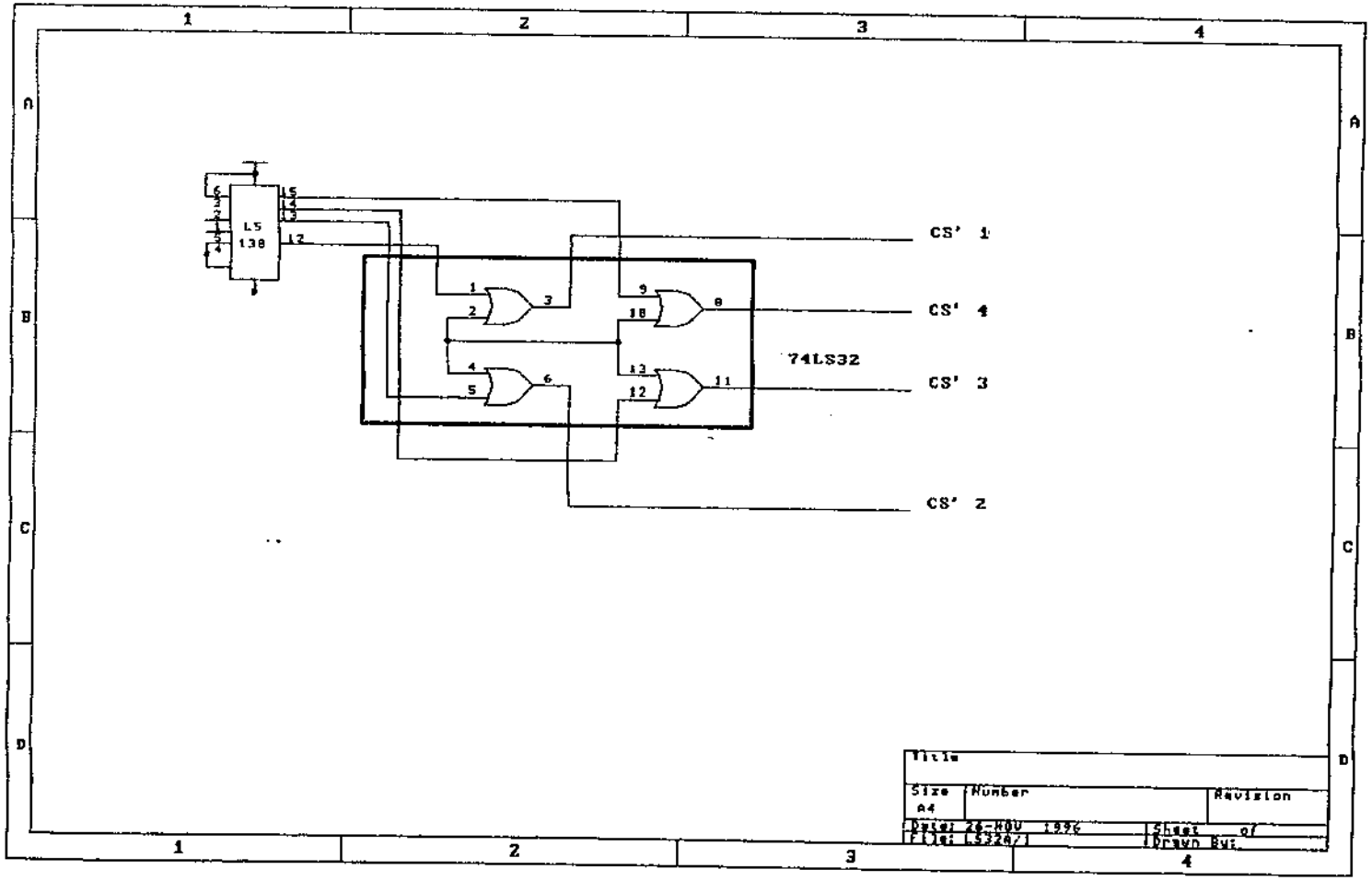
Tocci,1988, Digital Sytems , Prentice Hall Int pg 77

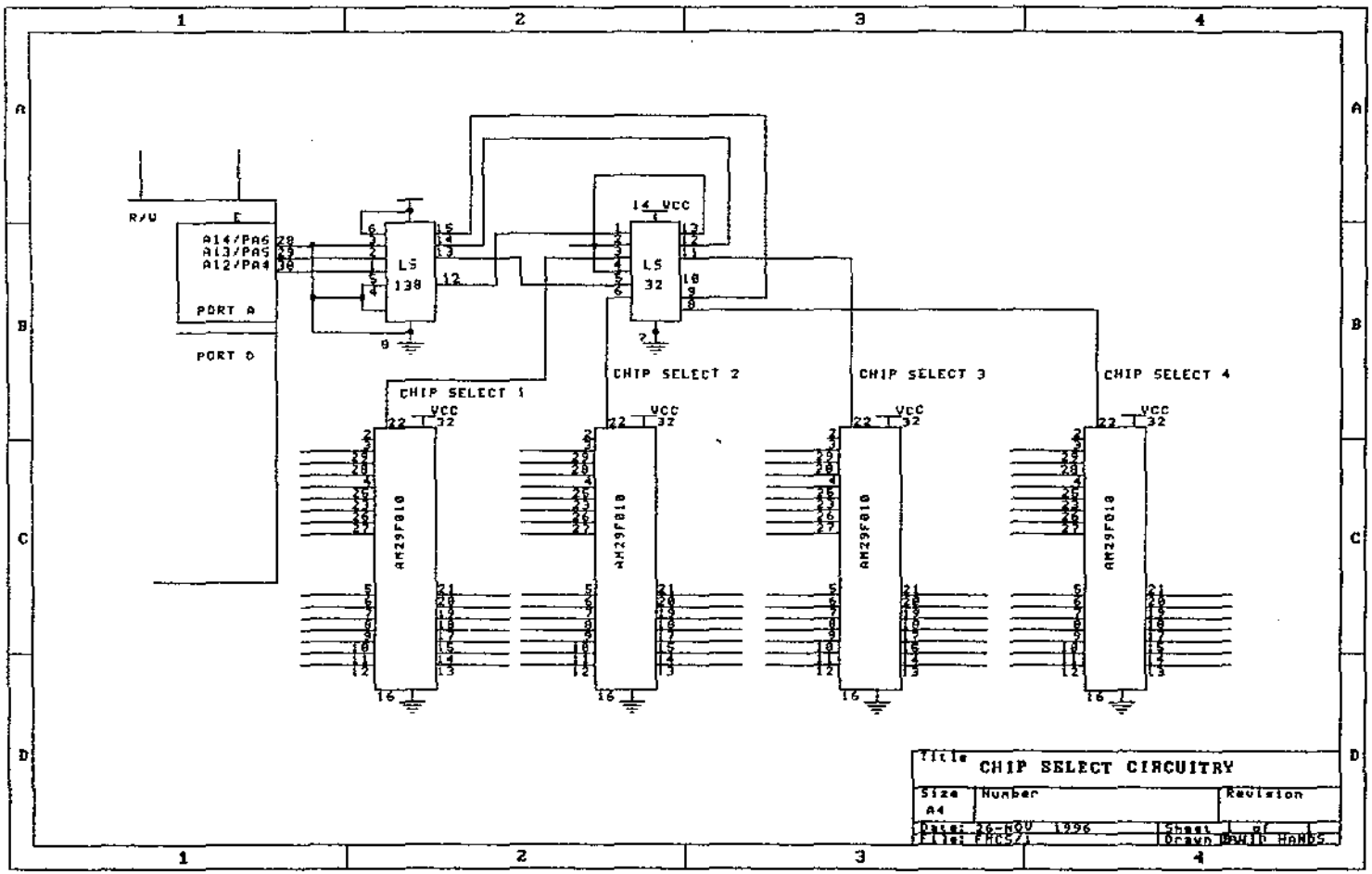
Windows Sources May 1994 v2 n5 p29, Ziff-Davis Publishing Company 1994

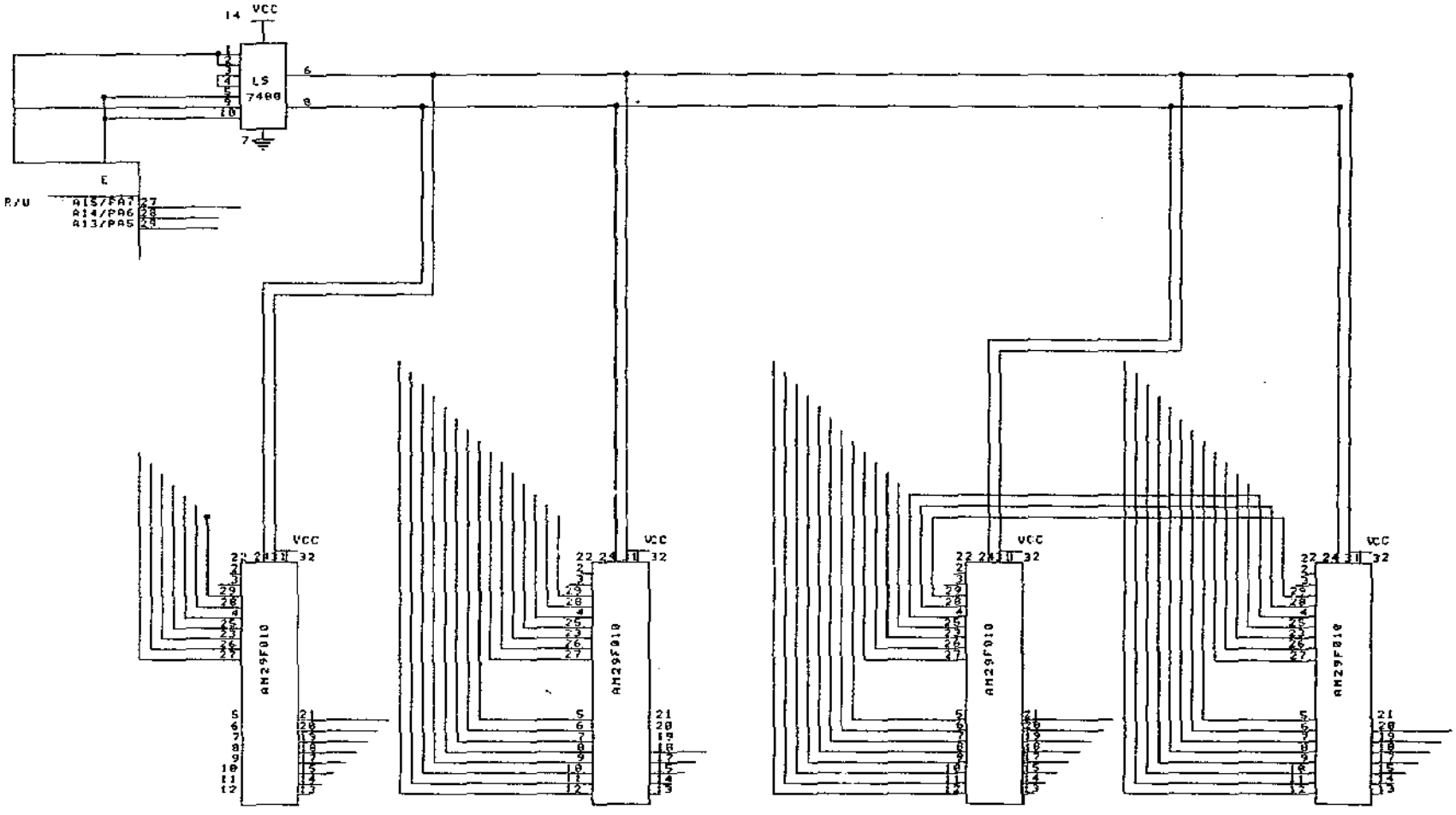
Walter, A. 1994, April 4 Motorola 68060 surfaces in Heurikon board. Electronic News v40 n2008 pg. 6 Electronic News Publishing Corporation.

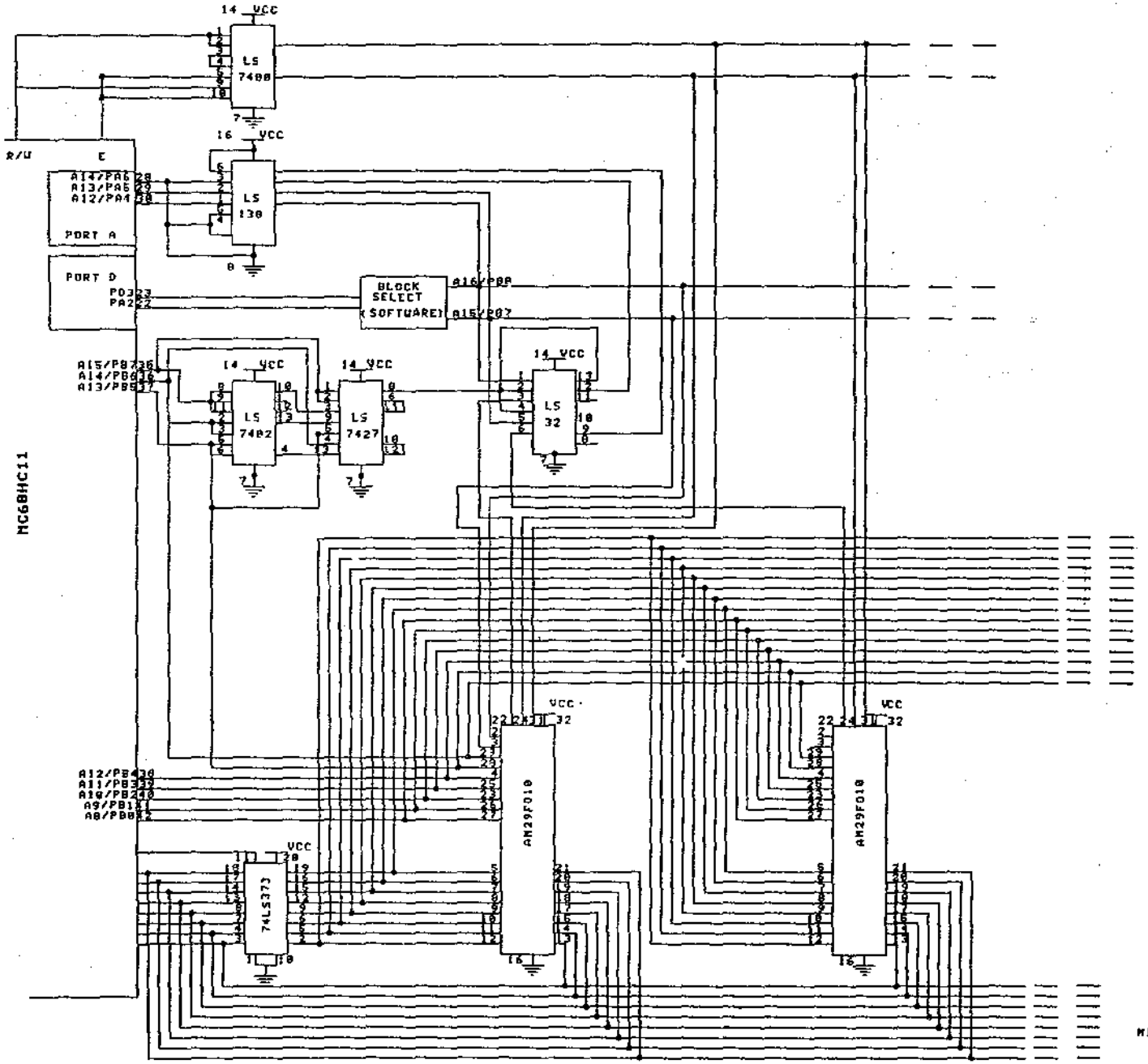

READ/WRITE SELECT
R/W'
E
PORT A
7
6
5
FM CHIP SELECT
4
3
BLOCK SELECT
A16
A15
MC68HC11
PORT B
A15
A14
A13
Flash Memory Select Logic
A8 - A14
AS
PORT C
ADDRESS LATCH SELECT
A16 CS OE WE
A15
FLASH-MEMORY 1
A14
A 8
DO - D7
A16 CS OE WE
A15
FLASH-MEMORY 2
A14
A8
DO - D7
A16 CS OE WE
A15
FLASH-MEMORY 3
A14
A8
DO - D7
A16 CS OE WE
A15
FLASH-MEMORY 4
A14
A8
D0 - D7
A0- - A7


MC68HC11
A14
A15
A13
VCC
GND
LS 7402
LS 7427
FM'
GND
FLASH MEMORY SELECT
Title MICROCONTROLLER-FLASH MEMORY PROJECT II 1998
Size
Number
LOGIC LS7402/7427
Revision
Date: 23 - OCT 1998
Sheet 1 of 1
File: LS7400\1
Drawn by: D Hands

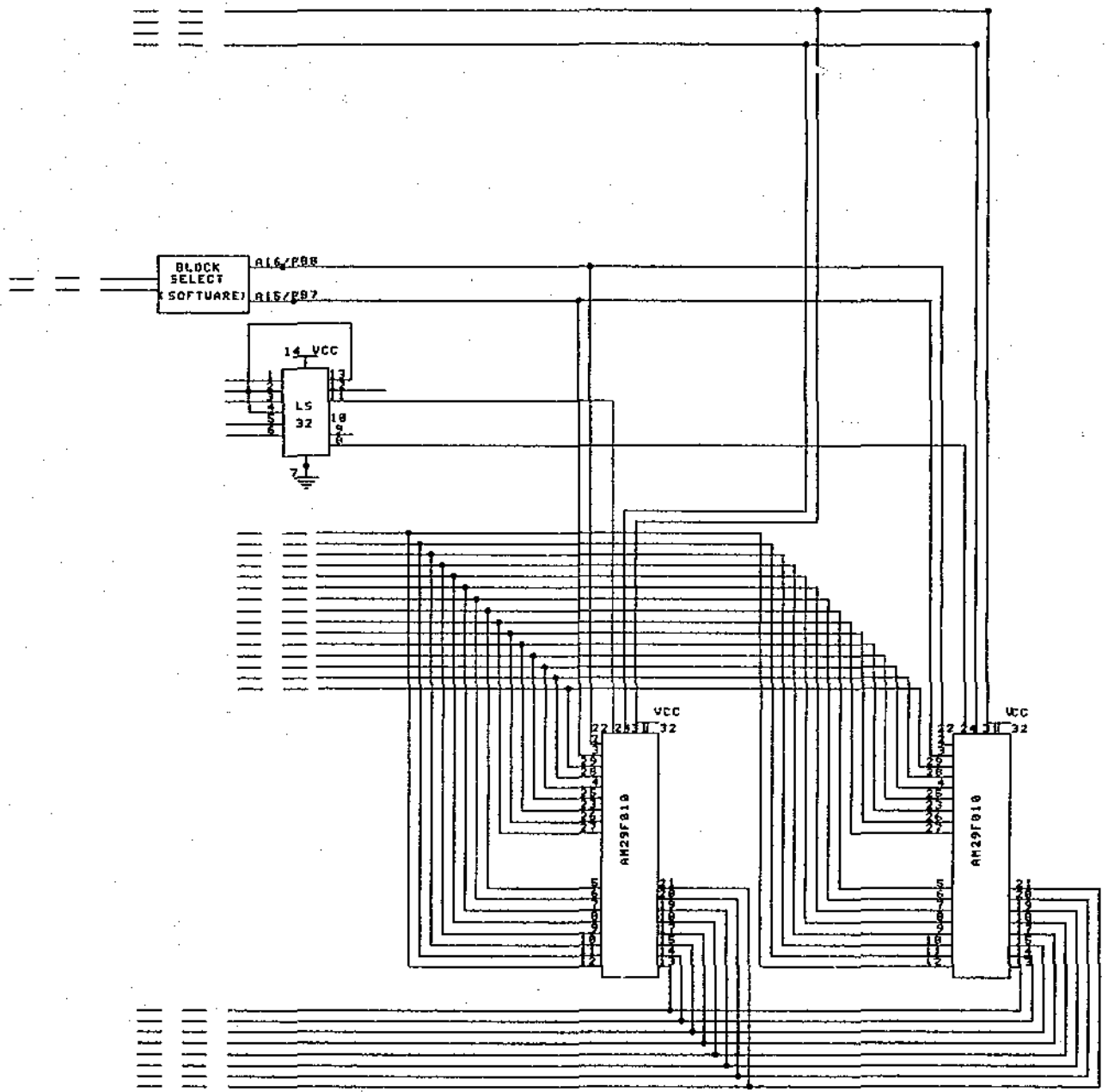

1
2
3
4
A
B
C
D
LS
138
74LS32
CS' 1
CS' 4
CS' 3
CS' 2
Title
Size
A4
Number
Revision
Date: 26-NOV 1996
Sheet of
File: (5224/)
Drawn By:


R/W
E
A14/PA6
A13/PA5
A12/PA4
PORT A
PORT D
LS 138
LS 32
14 VCC
CHIP SELECT 1
CHIP SELECT 2
CHIP SELECT 3
CHIP SELECT 4
VCC
AM29F010
Title CHIP SELECT CIRCUITRY
Size A4
Number
Revision
Date: 26-NOV 1996
Sheet
Drawn By: Hands


14
VCC
LS
7400
7
E
R/W
A15/PA7
A14/PA6
A13/PA5
AM29F010
AM29F010
AM29F010
AM29F010



MICROCONTROLLER-FLASH INTERFACE



MICROCONTROLLER-FLASH INTERFACE