4-BIT-SLICE MICROPROCESSORS IN COMPUTER FLIGHT SIMULATION

Ⓒ Nicolas Frangoulakis

A Major Technical Report

in

The Faculty of Engineering

Presented in Partial Fulfillment of the Requirements
for the degree of Master of Engineering at
Concordia University
Montreal, Quebec, Canada

March 1981

ABSTRACT

4-BIT-SLICE MICROPROCESSORS IN COMPUTER FLIGHT SIMULATION

Nicolas Frangoulakis

The objective of this major technical report is to examine the 4-Bit-Slice Microprocessors in Computer Flight Simulation. The report is divided into five chapters. In chapter one, the function of computer simulation and emulation is defined and the in-circuit-emulation facility is analysed. Also, the real time operation and the interface input/output system is studied. In chapter two, the architecture of the 4-bit-slice microprocessors is examined. In chapter three, the main computer and emulator system organization is dealt with in detail. In chapter four, the emulator microprocessor diagnostics are examined in respect to the interface input/output system. In the last chapter, the 4-bit-slice microprocessor is applied in the field of flight simulation. Its units are individually analyzed as well as its functions.

ACKNOWLEDGEMENTS

I would like to express my gratitude to Dr. David Gibbons, my adviser, for his support and guidance throughout the writing of this report. I would also like to thank the Canadian Avionics Electronics Ltd. for the practical and theoretical experience I have acquired during the last two years in their employment. Finally, I would like to acknowledge the help of my wife for the typing and proofreading of all the material.

LIST OF FIGURES

TABLE OF CONTENTS

INTRODUCTION

The purpose of this major technical report is to examine the application of the 4-bit-slice microprocessors in computer flight simulation. This application is one of the latest achievements in the evolution of microprocessor interfacing techniques. In the beginning when interfacing was not as complex and speed and flexibility requirements were not as necessary the interfacing methods were not so sophisticated. Today, however, and particularly in the area of computer flight simulation, the direction of evolution has lead towards the development of the third generation of microprocessors which provide an increased performance in speed and flexibility. The developed techniques range from interfacing the Central Processing Unit (CPU) to the Random Access Memory (RAM), Read Only Memory (ROM) and input/output devices to complete Floppy Disk Intelligent Interfacing. The interface begins at the wall socket and ends only at the front panel of the most remote peripheral. As the technology improves the interface designer will evolve into more of a programming person. And it is at this point where the hardware and the software areas interact to form what is referred to as firmware. It is this firmware application in the 4-bit-slice microprocessors that is dealt with in this report.

In the commercial world, a number of Large Scale Integrated (LSI) bipolar 4-bit-slice microprocessors are availabe such as the Advanced Micro

Devices AMD 2900 series, the Texas Instruments 75S481, the Motorola 10800 and the RCA 4057, to name a few. In this report, however, the bipolar 4-bit-slice AMD 2900 series microprocessor is studied and treated as a 16-bit microprocessor for industrial computer emulation applications with main emphasis in the field of computer flight simulation.

Simulation is the replacement of a real situation with a make-believe setting. Here the responses of the simulator function are very close to reality. However, the closer to reality the more expensive it becomes. In flight simulation the make-believe setting is represented by a plane cockpit. This make-believe plane cockpit has all of the built-in cues of a real plane which responds to simulating signals under the control of the main computer via the interface system. The main reason for building flight simulators is the complete elimination of all dangers involved during air crew training. Secondly, it is more economic to train personnel since the training cost only amounts to about 10% of the cost of a real plane usage. It is, therefore, because of this economic reason that there has been such a drastic increase in the application of flight simulators throughout the world since 1975.

Emulation is one of the applications of the bit-slice microprocessors and in effect it is a means to perform some or all of the instructions of the main system processor. In emulation the bit-slice microprocessor is used as a dedicated machine and performs tasks or routines on behalf of the main system computer, thus relieving the main processor and increasing the overall speed and capability of the system.

Bit-slice microprocessors have their own architectural Central Processing Unit (CPU) philosophy. In contrast to the single chip Metal Oxide Semiconductor (MOS) microprocessors, bit-slice microprocessors utilize two separate chips for the processing and the control function. This special structure of bit-slice microprocessors expands their functional capabilities by making them user-microprogrammable devices which are more suitable for specific design applications such as in computer flight simulation.

I. COMPUTER SIMULATION AND EMULATION

A. Emulator Microprocessor

The emulator microprocessor is a 16-bit microcomputer and its design resides in the 4-bit-slice Advanced Micro Devices AMD 2900 Family. Bit-slice microprocessors are manufactured with bipolar technology and are microprogrammable devices. They provide a variety of digital system architectures with various word lengths and instruction set capabilities, and increased effective speeds. Microprogrammability of the bit-slice microprocessors applies in computer emulation (flight simulation) where it replaces hardwired condition with software (programs), very effectively. The main disadvantage of bit-slice microprocessors is the system support software that is required because of the sophistication that the system operates on, since microinstructions are much more difficult to implement than macroinstruction (machine language) [1].

The emulator is studied here from a background point of view in both hardware and software and the term firmware will be applicable whenever the above two areas interact. The information in this chapter is based on the AMD 2900 Family Data Book and bit-slice microprocessor research papers, which are listed in the bibliography at the end of the report.

B. Computer Simulation

Simulation is the functional replacement of a hardware device by a program (software). The hardware and the software will generate the same outputs in response to the same inputs. The simulation performance (software), however, is much slower than the hardware and this is due to the fact that microinstructions require much longer to be implemented than hardwired instructions.

C. Computer Emulation

Emulation is simulation performed in real time. Emulators will simulate an operation much faster than the model's actual performance. For example, the bit-slice microprocessors will execute the instruction set of the main processor (being emulated) at the same or even higher speeds. During the program development stage, ROM emulation is performed by executing programs out of the RAM, as if they were stored in ROM. After the program in RAM has been tested (debugged), it is placed in a final ROM or PROM. There are several problems associated with this conversion. The main two problems are, the address conversion into the final ROM and synchronization whenever a slow RAM is replaced by a final faster ROM [2].

In emulation the microprocessor is basically used for two major functions. First to relieve the system's main processor by executing instrument update routines, function generation routines and other special functions. Second to provide the In-Circuit-Emulation testing facility routine which executes the system diagnostics (checks the microprocessor and the input/output Interface functions).

D. In-Circuit-Emulation

In-Circuit-Emulation is a facility that includes the system diagnostics (also called debugging). In any system where real input/output in real time must be tested, the major testing to be done is the emulation of the microprocessor itself. This is done by the In-Circuit-Emulator and this facility completely controls and tests the system under development, from the console. By using this software emulator it is possible to stop the operation of the microprocessor, examine the contents of the register or change them, examine the busses, the memory contents and even execute input/output instructions. Breakpoints are also provided by this facility which stop the program automatically at particular microaddresses (memory locations) during the execution of the program. If an error detection occurs at a breakpoint obviously it was caused by previous instructions and tracing is required to locate it. The in-circuit-emulator has a very important diagnostic capability for both software and hardware. It provides a checking tool for the complete system, that is to say the microprocessor and the input/output interfacing system [3].

E. Realtime Operation

The flight simulator system operates in a real time environment. This real time operation is accomplished by operating in a number of time frames which are controlled by the real time clock. This clock is normally run at a frequency of 20 Hz giving a frame time of 50 milliseconds. Every 50 milliseconds a real time clock interrupt causes a chain of events to occur:

a. Set up interface for input/output data exchange
b. Run simulator system programs
c. Set up instructor's station - updates, etc.
d. Run foreground mode operating systems
e. Run background mode operating systems

All of these events must occur within the time frame. This gives rise to one of the biggest problems in flight simulation which is running out of time. If the time frame is extended much beyond the 50 milliseconds, realism is then lost. When realism is lost, the response of the flight simulator elements (flight instruments, motion system, audio functions etc.) is partially or completely lost. One of the ways to overcome this problem is to run the programs at different rates which means that not all the programs are run every frame. The most important programs (critical) for example, the flight program, the motion program are run every frame. Other programs which are less important are run every other frame which means at half rate. Other programs are run every fourth frame (1/4 rate), others every eighth frame and so on. So that over a cycle of 16 frames every program will have been run at least once. The most important ones, of course, will have run 16 times. This 16 frame cycle is repeated continuously.

F. Direct Memory Access (DMA) and Programmed Input/Output (PIO) Transfers

The computer interface unit of the flight simulator is a DMA device and works on a cycle stealing basis. Once the interface unit gains control of the bus it can access the main computer memory at high speed without the intervention of the Central Processing Unit (CPU).

PIO transfer is the initialization process requiring address and control word information to be transferred from the main CPU to the computer interface unit (microprocessor). Every DMA transfer, however, requires a PIO transfer to initialize it.

G. Interface Input/Output (I/O) System

The interface I/O system is basically a vast digital to digital, digital to analog and analog to digital converter. It accepts the digital data from the main computer via the microprocessor system and converts it to the appropriate signals for the operation of the flight simulator and vice versa. The microprocessor is the master controller of the interface I/O system containing the memory, the processors necessary to convert the computer data format into interface format, address information, error checking devices, and controllers for communication with the I/O interface system chassis. The Interface I/O System consists basically of the following two units:

a. Sub controllers
b. I/O cards

The subcontrollers are located in the I/O interface chassis and each controls a number of dedicated system I/O cards. Each subcontroller has a unique address. The I/O cards contain a number of input/output devices (words) which connect to components in the flight simulator (cockpit).

II. 4-BIT-SLICE MICROPROCESSORS

A. Architecture of the 4-bit-slice microprocessor

The Central Processing Unit (CPU) of the bit-slice microprocessors is a multi-chip implementation and is divided into two sections, the CONTROL section and the PROCESSING section as shown in Figure 1.

The control section consists of microprogram memory, a microprogram sequencer, selection logic and pipeline registers all of those being implemented on separate LSI chips. The microprogram memory (ROM or PROM) contains the microinstructions required for the control of the parallel operation of the RALU (Register Arithmetic Logic Unit) slices,

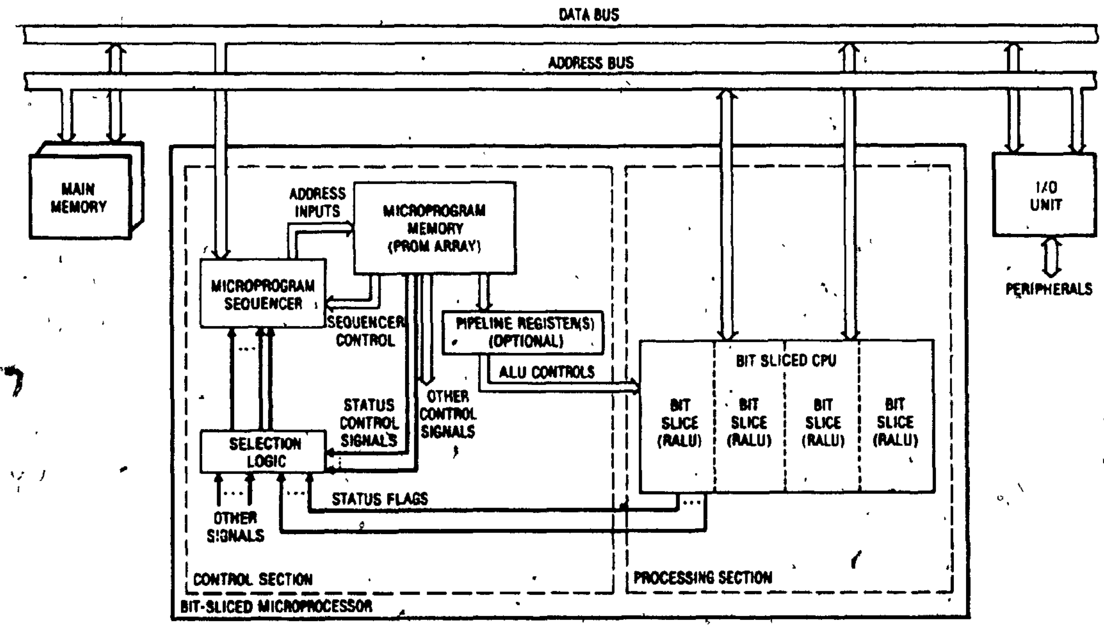


Figure 1. A Microprogrammable Bit-Sliced Microcomputer

in the processing section. The microprogram sequencer decodes the macroinstructions or sequences the microinstructions and generates the next microaddress. The selection logic accepts status flags and other control signals and decodes them to feed the microprogram sequencer. The pipeline registers are inserted between the microprogram memory and the RALUs to overlap the present microinstruction execution with the fetching of the next microinstruction and thus provide speed improvement (Figure 1) [4].

The processing section carries out the arithmetic and logic operations and consists of four 4-bit-slice RALUs. Each RALU contains registers, one accumulator, one arithmetic logic unit and status flags. The slices operate in parallel (cascaded) and can thus handle various word lengths, such as 24, 32 and even 48 bits, which are multiples of the basic 4-bit slice.

During the control operation, the microprocessor fetches macroinstrucions from the main system memory. This fetching process is accomplished under a READ microinstruction residing in the microprogram memory. The microprogram sequencer in turn interprets the operation code of the macroinstruction and executes it as a series of microinstructions. The OPERAND part of the macroinstruction is sent to RALU for the computations and main memory address manipulation (Figure 1) [1].

B. The Advanced Microdevice AMD2900 Microprocessor

The 4-bit-slice bipolar microprocessor is a cascadable element

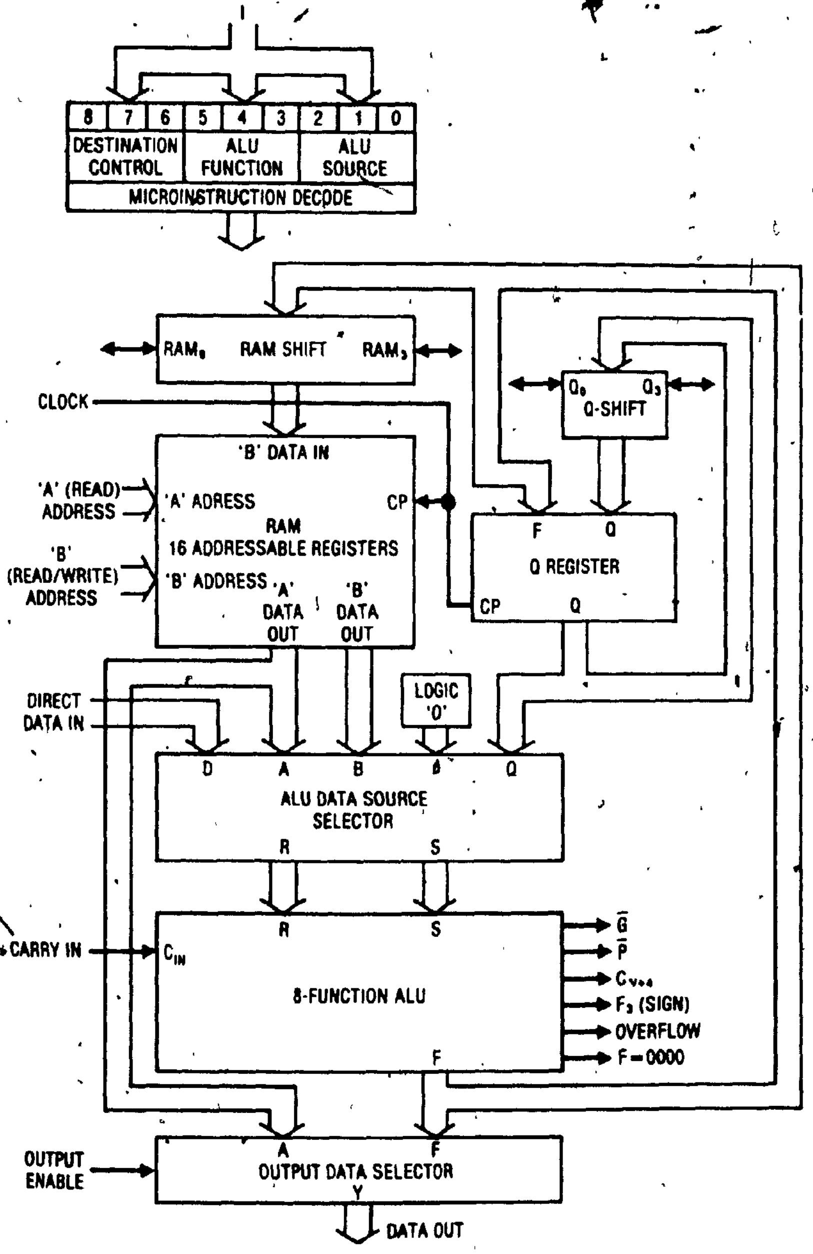
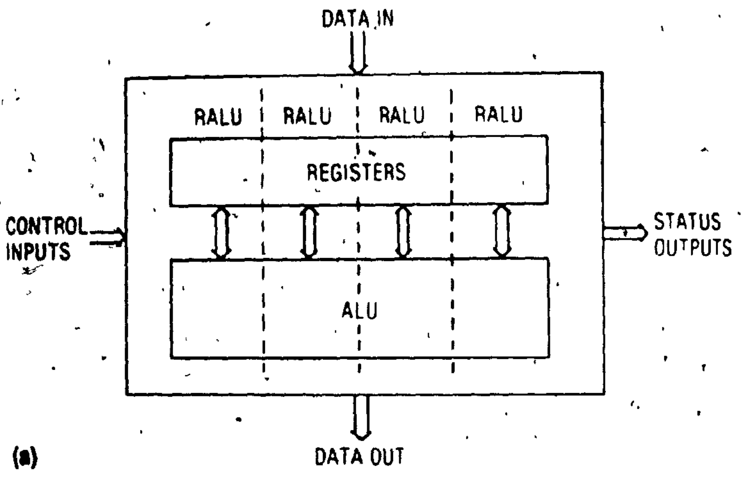


Figure 2. Block diagram of comercially available bit-sliced microprocessor AM2901 4-bit RALU (from The AM2900 Family Data Book p. 2-3)

with a high operating speed. The flexibility of this device allows efficient emulation of a variety of digital machines. The AM2901 4-bit bipolar microprocessor is shown in Figure 2. The device consists of a 16-word by 4-bit two port RAM, a high-speed ALU, and the associated shifting, decoding and multiplexing circuitry. The Instruction Register (IR) accepts a 9-bit microinstruction word which organizes into three groups of 3 bits each, the ALU SOURCE OPERANDS, the ALU FUNCTION and the ALU DESTINATION register. The microprocessor is cascadable with full Look-Ahead-Carry (LAC) or RIPPLE Carry, and provides various status flags outputs from the ALU [1].

The processing section is a vertical partitioning of the CPU. Such a partitioning, slices the registers and the ALU into equal-length and functionally equivalent parts, called Register Arithmetic Logic Units (RALU) or bit-slices (Figure 3). Each RALU handles 4 bits and can be cascaded to process a variety of lengths. The arithmetic operations and the sources and destination for the ALU are the same for all slices. The input data bus is divided into proper length sections (enter the slices), and the output data is recombined when existing the slices. A RALU slice contains and ALU, a multiple word register file, a shifter, data I/O lines, control inputs and status bit outputs.

The arithmetic and logic operations are performed by the Arithmetic Logic Unit (ALU), which constitutes the core of the processing section. The ALU allows arithmetic and logic operations at desired word lengths. It also includes an internal accumulator and a shift matrix as well as internal registers used as temporary storage for the function of multiplication and an extension register for double shift results. Additional features of the ALU allow detection of overflow, zero, sign conditions and Look-



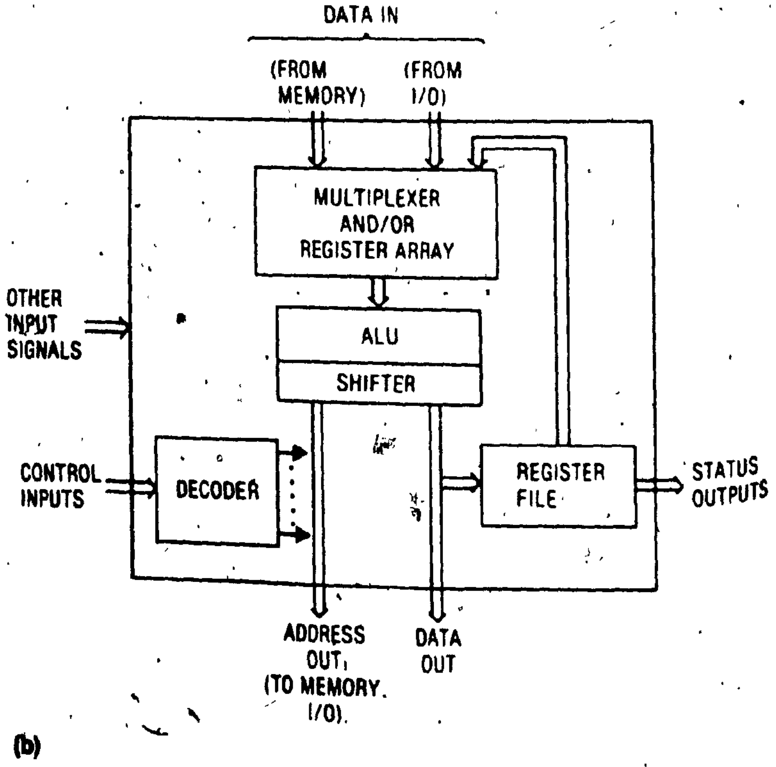


Figure 3. (a) A vertically partitioned processing section defining the RALUs and (b) General block diagram of a RALU

Ahead-Carry (LAC) performance. All arithmetic and logic functions performed by the bit slices are dictated by the microprogram memory and the microprogram sequencer. The sequencer decodes the operation code of a macroinstruction fetched from the main memory and provides the microprogram memory with a new address from which the microinstructions of the new macroinstruction will be fetched. Each microinstruction coming out of the microprogram memory causes the appropriate commands (control signals) to be sent to the bit-sliced processing section of the microprocessor [4].

C. Microprogrammability of the Bit-Slice Microprocessors

(1) Evolution: Microprogramming evolved in four phases or generations with the first phase starting in the early 1950's, when diode matrix technology was used for the microprogram memory. The second generation involved magnetic core memories and started in the late 1950's. The third phase started in the lare 1960's when the bipolar monolithic control memories appeared. The fourth generation or current phase started in 1974-75 when the programmable Large Scale Integrated (LSI) devices appeared. This new phase lead to the 4-bit-sliced microprocessors with parallel bit-slice operation which allowed various word length implementation, higher speeds and system design flexibility at relative inexpensive rates.

(2) Microprogram Sequencer: The purpose of the microprogram sequencer is to present an address to the microprogram memory so that a microinstruction may be fetched and executed. The next address logic part of the sequencer determines the specific address source to be loaded

into the microprogram address register/counter. This feature of the next address generation improves overall system speed performance. To ensure microprogrammability the microprogram sequencer (control section) should be able to implement the basic control structures as shown in Figure 4.

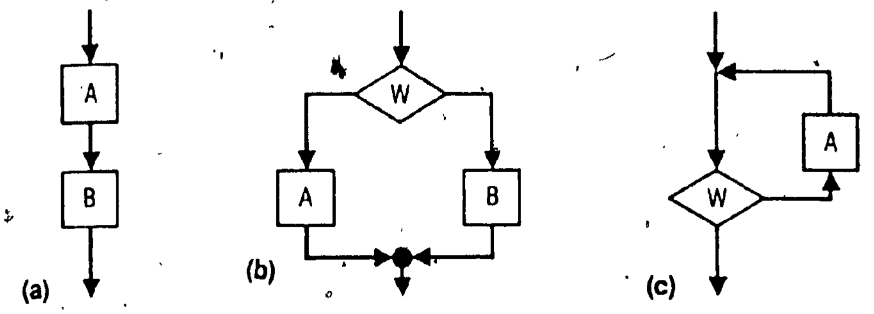


Figure 4. Basic control structures implemented in microprograms: (a) A then B, (b) if W then A else B, and (c) while W do A (where A and B represent micropragram modules)

(3) Microprogram Memory: The microprogram memory of the bit-slice microprocessor contains sequences and functions called microinstructions. A single microprogram memory word (a microinstruction) may be as much as 100 bits long while a microprogram may consist of 1000 words or more [5].

D. The Advanced Micro Devices 2909/2911 Microprogram Sequencer

(1) The AMD Microsequencer: The Advanced Micro Devices 2909/2911 bipolar microsequencers are used in high-speed microprocessor applications. This is a 4-bit-slice device and is cascadable to allow extensive addressing of microprogram words as shown in Figure 5. The

device contains a four-input multiplexer used to select either the address register, direct inputs, microprogram counter, or file as the source of the next microinstruction address. The multiplexer is controlled by SØ and S1 inputs. The address register consits of four D-type edge triggered flip-flops with a common clock enable. The address register is available at the multiplexer as a source for the next microinstruction address. The direct

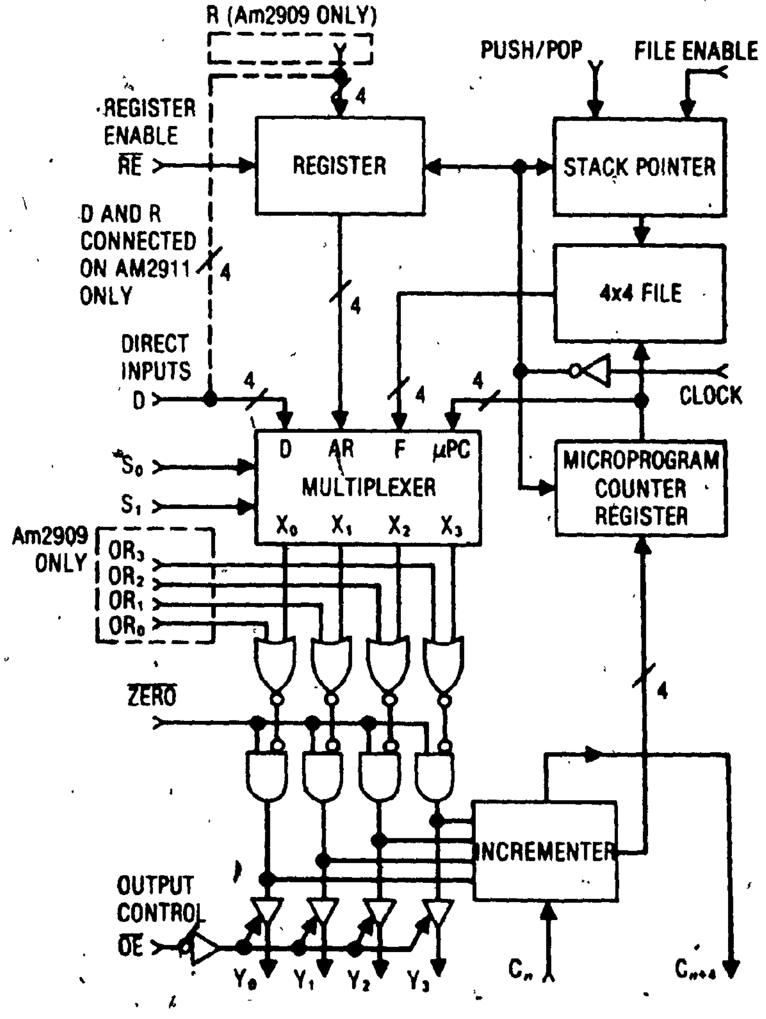


Figure 5. The AM2909/2911 LSI Microprogram Sequencer (from the AM2900 Family Data Book p. 2-74)

input is a four-bit field of inputs to the multiplexer and can be selected as the next microinstruction address [4].

The microprogram counter is a 4-bit incrementer, basically, followed by a 4-bit register. The incrementer has Carry-in (C_n) and Carry-out (C_{n+4}) such that cascading to larger word lengths is easily applicable. The 4 x 4 stock file is also applied to the multiplexer. The file provides return address linkage when executing microsubroutines. The file contains a built-in Stack Pointer (SP) which always points to the last file word written. This allows stack reference operation (looping) to be performed without a push or pop. The stack pointer operates as an up/down counter with separate PUSH/POP and file enable inputs. The stack pointer linkage is such that any combination of pushes, pops or stack references can be achieved. One microinstruction subroutine can be performed and since the stack is 4 words deep, up to four microsubroutines can be NESTED. The ZERO input is used to force the four outputs to the binary zero state. Each Y output bit also has a separate OR input such that a conditional logic can be forced at each Y output. This allows JUMPING to different microinstructions or programmed conditions. To conclude, the AM2909/2911 microsequencer is a four-bit wide address controller intended for sequencing through a series of microinstructions contained in a ROM or PROM. The AM2909/2911 can select an address from any of the following four sources:

a. A set of external direct inputs (D)
b. External data from the R inputs, stored in an internal register
c. A four-word deep PUSH/POP stack
d. A program counter register which contains the last address plus one

Each of the four outputs can be OR'ed with an external input for conditional SKIP or BRANCH instructions.

The AM2909/2911 receives the above inputs and specifies one of the following outputs:

a. Increment
b. Conditional SKIP next instruction
c. Conditional BRANCH to a microsubroutine
d. PUSH/POP the register stack, and so on.

For further detailed information concerning microprogram sequencers, microprogram controllers, priority interrupt controllers and other microprocessor system elements, reference may be made to the Advanced Micro Devices, The AM2900 Family Data Book [4].

(2) The AMD Look Ahead Carry (LAC): A 16-bit microprocessor is formed by interconnecting 4-bit slices (RALUs) to allow arithmetic and logical carries between chips (figure 3). These RALUs with the aid of circuity such as Look Ahead Carry (LAC) generation and buffering form the operator section of a microprocessor. The AM2902 LAC is a high-speed, look-ahead carry generator which accepts up to four pairs of carry propagate and carry generate signals and a carry input and provides anticipated carries across four groups of binary ALU's. In this manner the carry is sensed before the actual operation (addition) is executed thus saving overall time performance.

III. MAIN COMPUTER/EMULATOR SYSTEM ORGANIZATION

A. General

The main computer is a minicomputer system and can be either a Digital Equipment Corporation (DEC), a Texas Instruments (TI) 980 or any other minicomputer system available in the market. The main computer performs all the calculations and stores the results in its memory. The microprocessor system then takes the output from the main computer and sends it to the interface I/O system which converts the data to the appropriate signal levels. The flow of the input data is simply the reverse which means that signals from the flight compartment are converted to data by the I/O interface, the microprocessor takes this data and sends it to the main processor where it can be processed.

The microprocessor is designed and built as a 16-bit microcomputer and its functions are as follows:

a. To control data transfers between the main computer and the interface I/O system
b. To provide the function generation routine
c. To provide the instrument scaling routine
d. To perform bytes to bits conversion and vice versa
e. To perform the system diagnostics

The data transfer function is done by having a table of all inputs and outputs stored in the microprocessor memory. Whenever a change between the table and interface inputs is detected, the new input is sent to the main computer memory and conversely for outputs. Every input and output corresponds to an interface I/O assignment, which is listed in the Cross-Reference (XREF) area of the main computer memory. In this area are stored all those variables which are used by more than one program. For example, the electrical program will store a 16 bit word in this area to show the simulated bus power availability. Other programs will look at the state of this word to determine its outputs to the system. When the cross-reference is compiled special tables are created for the microprocessor relating the labels with the corresponding interface I/O assignments. When starting up the simulator, these tables are sent from the main computer system memory to the microprocessor memory [6].

Function generation is merely two or three dimensional data interpolations. ~~To be~~ appropriately used the original data must be processed to a form usable by the microprocessor. Once the necessary data file have been produced they are loaded at startup of the simulator. In the XREF there are areas of memory set aside for input to the microprocessor function generation program. The user program requests that the function generation be done by setting a flag in the XREF. The microprocessor detects this executes the routine and then resets the flag once the function generation is complete.

Instrument scaling involves the conversion of output data in engineering units to a form usable by the I/O interface system (say between Ø and 1). Thus for a temperature indicator the program output could be 98°C and the microprocessor will make the correct conversion to cause the dial to receive a signal deflecting it to 98°C on its scale. The user inputs calibration points form a file and are loaded into the microprocessor memory at startup.

Discrete outputs and inputs (i.e. lights and switches) are usually stored as bytes in the main computer memory. However, the interface needs only one bit to identity an ON/OFF condition. For this reason the microprocessor converts all bytes from the main computer to bits and vice versa storing, transmitting and receiving all discrete variables as bits to and from the interface I/O system. The system diagnostics are performed by the microprocessor and are dealt with more extensively in Chapter IV.

B. Memory Organization

(1) Bootstrap: The microprocessor memory is organized as shown in Figure 6. The top of the memory block diagram is occupied by the PROM which ranges from 0000 to 07FF. In this PROM reside the emulator system Bootstrap and diagnostic routine. When the system is switched on the microprocessor will automatically boot and then run its memory diagnostics [2].

(2) Program Code: After the memory test is successfully done then the main system processor will load a buffer with the Program Code. The Program Code is then transferred into the microprocessor memory within the 32K RAM (occupies 8K) at an address location between 4040 and 6000. The Program Code loading into the microprocessor memory is done by executing the LOAD CODE interrupt service command. The Program Code loading starts at location 4040 which is hardwired into the RAM. The LOAD CODE

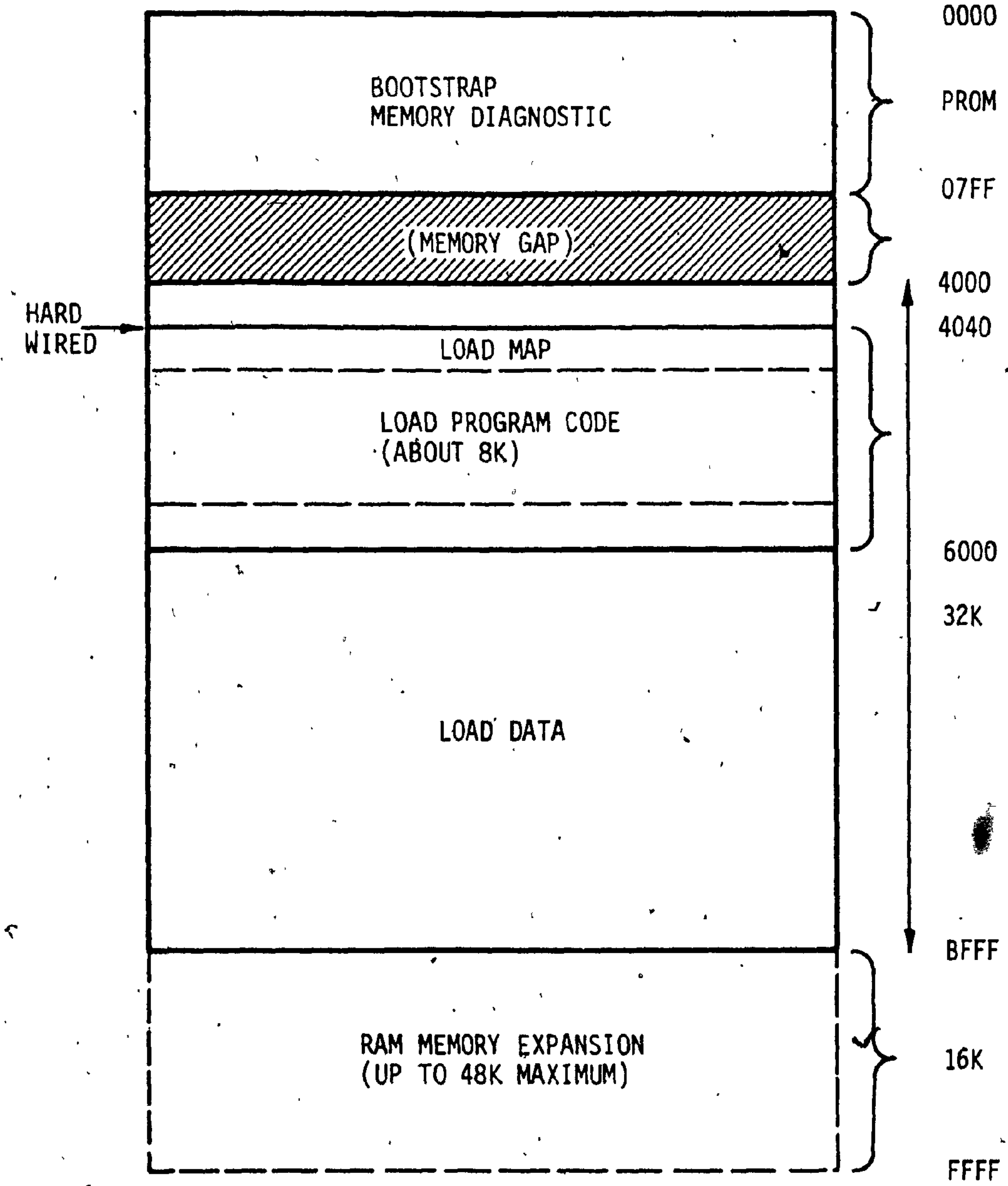
command is a main computer interrupt and provides information about the address where the main computer information resides and the size of the buffer that contains that information (word count).

(3) Load Data: After the Program Code has been loaded the data will be loaded into the microprocessor memory by executing the LOAD DATA main computer interrupt service. The data is loaded starting at address memory location 6000 into the 32K RAM, and contains information about all the location of the simulator hardware. The LOAD DATA command provides information about where the information resides (in the main computer) and what is the length of the buffer (word count) that contains it. The data loaded contains information about the discrete outputs, and inputs, analog outputs and inputs and synchro outputs and inputs which correspond to the input/output Interface system.

(4) Load Map: Finally the LOAD MAP command (main computer interrupt) complements the LOAD DATA command by providing input/output mapping information (where the interface I/O are mapped). The LOAD MAP command when executed provides to the microprocessor three elements of information:

a. The key which provides the type of I/O interface assignments
b. The starting point of the main computer buffer that contains the I/O interface information
c. The length of the main computer buffer that contains the I/O interface information

After the LOAD MAP is complete all the information that the emulator needs has been provided and it can start running. The START command informs the microprocessor to start executing its main program routines at a particular location. A STOP command will stop the program running, while a CONTINUE command will continue the program from where it stopped [2].



BOOTSTRAP MEMORY DIAGNOSTIC
(MEMORY GAP)
HARD WIRED
LOAD MAP
LOAD PROGRAM CODE (ABOUT 8K)
LOAD DATA
RAM MEMORY EXPANSION (UP TO 48K MAXIMUM)
0000
PROM
07FF
4000
4040
6000
32K
BFFF
16K
FFFF

Figure 6. Microprocessor System Memory Organization

C. Interrupt Services

(1) General Interrupts: The emulator test utility allows access and control of the Microprocessor/Memory module by the main system processor. For example, to check if the microprocessor is running we can read a particular location in the microprocessor memory. This reading gives information about the microprocessor being idle or executing its main program. A HELP file is available with about 40 commands some of which are:

RESET - This command is a hardware reset to the microprocessor. It enables the Microprocessor to execute its Power-On-Boot-Strap Diagnostics, and waits for a command from the main processor.

READ - This command reads the contents of the specified microprocessor memory address and displays it. The microprocessor memory is organized in two parts. The first several 2K words (0000 to 07FF) are reserved for storing the ROM and the next 32K words (4000 to BFFF) are used for the RAM storage (code, data and mapping).

WRITE - This command writes specified value into the microprocessor memory specified location.

LOAD CODE - This command loads the microprocessor main program from a specified file in the main system computer.

LOAD DATA - This command loads the microprocessor with data tables that describe the interfacing input/output functions such as instrument scaling, function generation etc.

LOAD MAP - This command loads the required main computer registers to enable the microprocessor to READ and WRITE to a specified main computer memory location. There are usually a number of buffers loaded (mapped) into the microprocessor from the main computer. Some of these buffers map:

a. Discrete Inputs
b. Discrete Outputs
c. Analog Inputs
d. Analog Outputs
e. Synchro Outputs
f. Word Inputs
g. Word Outputs
h. Random Number Generations
i. Special Function Requests

Boot - This command reloads the microprocessor when it stops without having to shutdown and restart the whole procedure. The boot command simply executes consecutively the following commands:

a. RESET
b. LOAD CODE
c. LOAD DATA
d. LOAD MAP

Other commands are available in the HELP file such as START, STOP, CONTINUE, BLOCK READ, BLOCK WRITE, QUIT, EXIT, etc. These commands can be requested in the microprocessor test utility routine and allow access to the microprocessor memory by the main system processor [6].

(2) Emulator Diagnostic Interrupts: The emulator diagnostic interrupts concern the microprocessor internal operation and have nothing to do with the main system processor interrupts. The CLOCK interrupt resides in the microprocessor and executes a variety of functions. It is composed of a programmable clock timer which runs in multiples of 62.5

microseconds, which is derived from the basic clock frequency. Normally the clock runs in 5 millisecond intervals (62.5 x 80 = 5 msec), for the purpose of servicing the following special request functions (Figure 7):

a. Function Generation Routines
b. Poppable Circuit Breaker Routine
c. Serial Navigational Routine
d. Record and Play Back Routine

The clock also runs in 25 milliseconds (every fifth clock interrupt) intervals for synchro instrument updating. This is a service routine of high priority because synchro instruments tend to step down if they are not refreshed frequently. Another function (watchdog) is done by running the clock in 100 millisecond intervals. The purpose of this function is that whenever the simulator is running the executive program (main computer) is incrementing a counter. This counter is monitored by the microprocessor which recognizes running or non-running simulator status. The emulator will reset itself if the main processor stops running.

D. Timing Sequence for Main Computer-Emulator-I/O Interface

The timing sequence for the update of the simulator interface system input/output modules is shown in Figure 7. The time for a complete update is approximately 50 milliseconds and this will depend on the number of input/output modules used in the interface I/O system. As soon as the updating sequence is completed the cycle will restart from the discrete inputs. The timing sequence updates the following input/output interface functions:

a. Discrete Inputs (DI's) - are transferred from the interface I/O system to the main computer (via the microprocessor)

b. Word Inputs (WI's) - are transferred from the interface I/O system to the main computer directly (without being manipulated, via the microprocessor)

c. Analog Inputs (AI's) - are transferred from the interface I/O system to the main computer (via the microprocessor)

d. Discrete Outputs (DO's) - are transferred from the main computer to the interface I/O system (via the microprocessor)

e. Analog Outputs (AO's) - are transferred from the main computer to the interface I/O system (via the microprocessor)

f. Word Outputs (WO's) - are transferred from the main computer to the interface I/o system directly (without being manipulated, via the microprocessor)

g. Synchro Outputs (SNOPS)- are tansferred from the main computer to the I/O system (via the microprocessor)

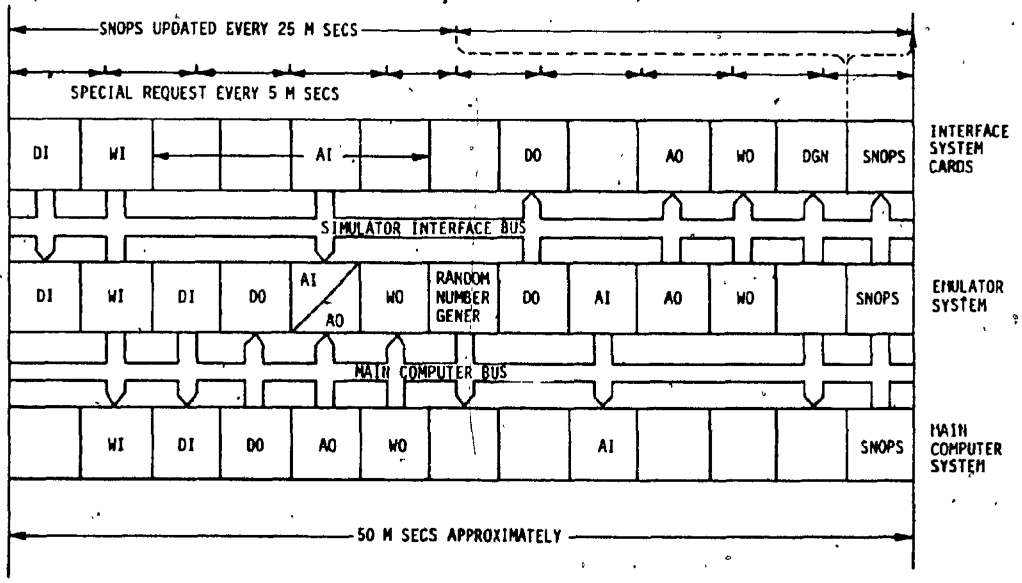


Figure 7. Timing upadate sequence for the Interface System Input/Output Modules

IV. EMULATOR INTERFACE DIAGNOSTICS

A. General

The Emulator Interface Diagnostics are exercised as an on-line system diagnostic routine also called the Interface System Status. This routine is run by the microprocessor which checks periodically the inter-face system modules for address and data integrity. The errors are en-coded into messages and sent to a buffer in the main computer where they are formatted and logged. The most common interface system status errors are the General Transfer (GT), Parity (P), Address Time Out (ATO), Data Time Out (DTO), Chassis Power Supply Status (CPSS) and Input Output (I/O) module functional integrity error. Each of the above Interface System Status errors correspond to a bit setting in a status register in the Simulator Interface Controller (SIC) module. The microprocessor scans the status register periodically and informs the main processor about the error being set. The main processor inturn, logs the error and prints it on the computer system terminal [7]. The following I/O modules are checked by the on-line system diagnostic routine:

a. Discrete output module validity
b. Discrete input module validity
c. Analog output module accuracy
d. Analog input module accuracy

The information in this chapter has been derived from my personal experience in working in the field of computer system simulation.

B. General Transfer (GT) Error

The general transfer error specifies that transfer of data through either the Main Computer Interface Controller (MCIC) module or through the Simulator Interface Controller (SIC) module is not successful. This is a general emulator error and the cause could be traced anywhere in the micro-computer system. The general transfer error is a global error and therefore, it cannot be masked or bypassed since it causes the overall operation of the simulator system to stall.

C. Parity (P) Error

The parity error may occur on the address word or on the data word and odd or even parity can be checked. Parity error arises when the microprocessor reads data from the interface system input/output modules or from the memory module. Whenever information is stored a parity bit (the seventeenth) is also stored, and when information is read back a parity bit is also read and compared with the stored parity bit, as a parity checking error. The parity error is a channel error which can be bypassed and therefore allows the overall operation of the simulator system to continue.

D. Address Time Out (ATO) Error

The address time out error is set if an interface input/output module does not respond to asserted address word by the microprocessor. The microprocessor waits for a few milliseconds for the address synchroni-zation signal which is asserted by the interface. However, if the interface response does not occur, the address time out flip-flop will be set thus producing an ATO error.

E. Data Time Out (DTO) Error

The data time out error is set if an interface input/output module does not respond to asserted data word by the microprocessor after normal address word response. The microprocessor waits for a few milliseconds for the data synchronization which is asserted by the interface; and if the interface response does not occur the data time out flip-flop will be set thus producing a DTO error.

F. Power Supply Status (PSS) Error

The power supply status error occurs as a result of monitoring the 5 Vdc logic, the ±15 Vdc and the ±24 Vdc of the interface input/output modules, whenever an overvoltage occurs. The power supply status routine is exercised by setting an address word control bit and reading back to the microprocessor the address word as normal data. Thus the microprocessor scans (reads) the address word and if a bit is set an overvoltage situations is shown. The power supply status error hardware function is done by the sub controller modules, which perform the bus interfacing function for each individual chassis while the software routine resides in the microprocessor [8].

G. Discrete Output Validity

On each discrete output (DO) module the output data is latched (the voltage across the relay coil) by a second data latch. The data is read back to the microprocessor by issuing a Data Request (DR) command after the appropriate addressing. This command causes the 16 bits of data in the second latch to be sent (serially) to the Diagnostic (DGN) module, from which they are returned to the microprocessor (in parallel). If there is no error on the discrete output module then the data received should correspond exactly with the last output data sent to the card. The flow chart of the Discrete Output Validity is shown in Figure 8.

H. Discrete Input Validity

The discrete input (DI) module is read back by the microprocessor in both normal and inverted mode. These two readings are enabled when the microprocessor sets control bit of the address word appropriately. The two readings are compared via an exclusive "OR" and checked for all ones (high). For correct discreet input validity the two readings should be separated by a few microseconds not to allow any real time setting which would result in a faulty error detection. The flow chart of the Discrete Input Validity is shown in Figure 9.

I. Analog Output Accuracy

On each analog output (AO) module the output data is latched into a holding register. The output of this register goes through a D/A converter to drive the output. This output signal is monitored by issuing the appropriate Interface input/output Bus address and a Data Request (DR) command. The analog voltage is then passed along the chassis backplane into a holding register by the Data Available (DA) signal from the Interface input/output Bus. The output of the holding register drives the output relay through a Darlington pair. It is this signal which is monitored

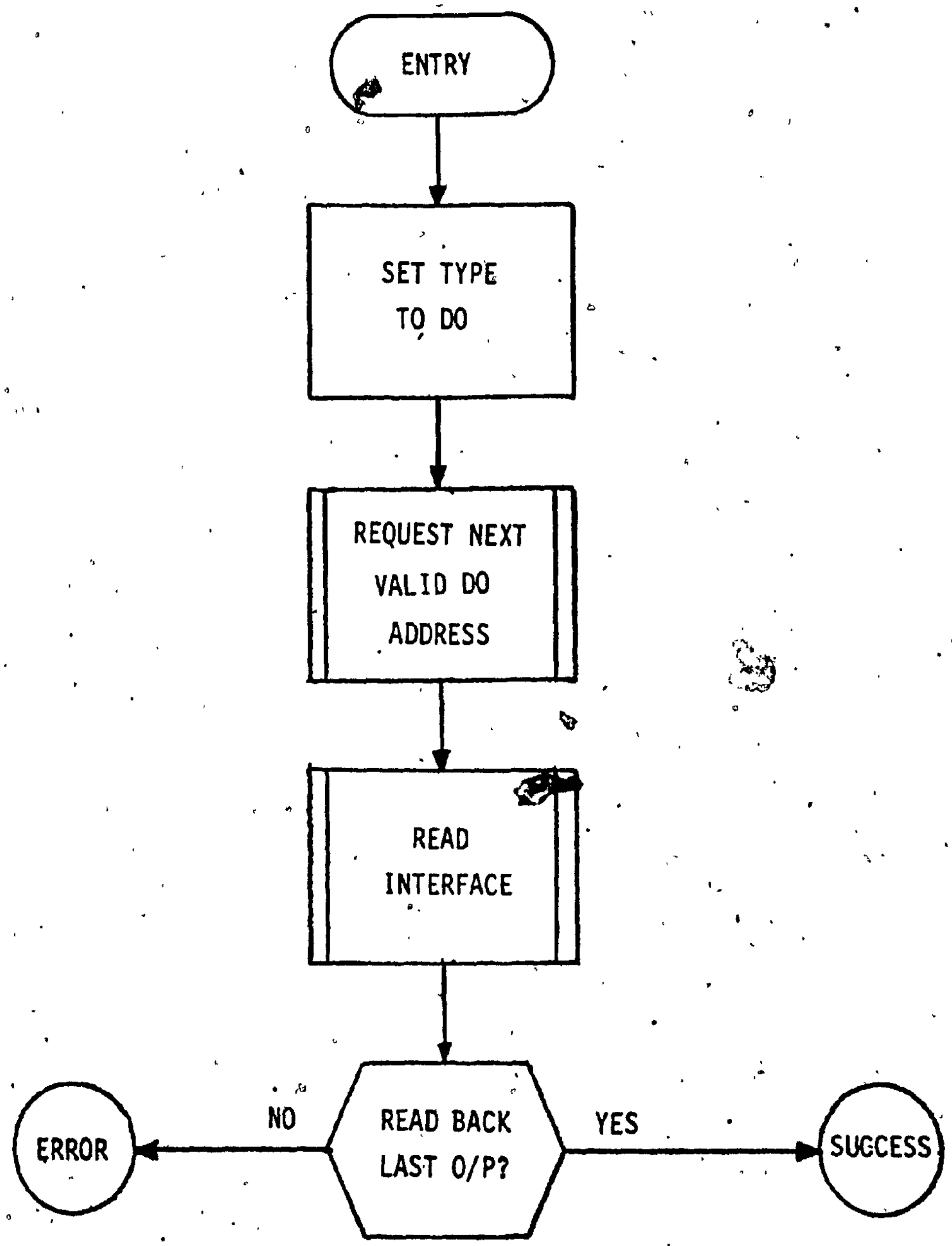


Figure 8. Discrete Output (DO) Validity Flow Chart

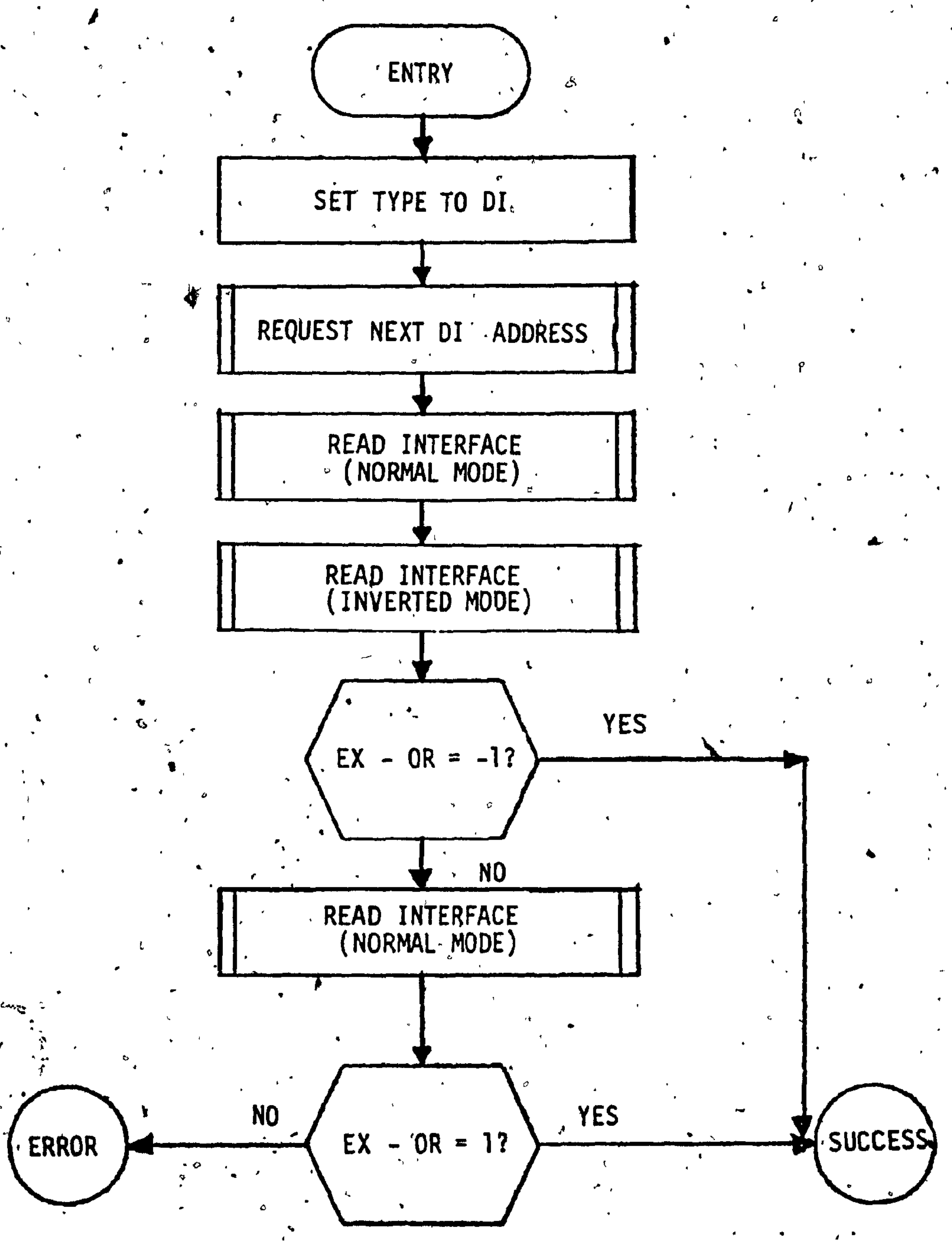


Figure 9. Discrete Input (DI) Validity Flow Chart

to the Diagnostic (DGN) module where it is converted back to a 12-bit discrete value via an A/D converter. This value is read back to the microprocessor and compared with the last output value. Due to the limited slew rate of the analog output, the following check is performed:

The check fails only if the read back value, V_i, falls outside the limits $AOMIN_i$, for 10 consecutive samples where:

$$AOMIN_i = MIN\ (AOP_j - LIM) \qquad j = i, \ldots i$$

$$AOMAX_i = MAX\ (AOP_j + LIM) \qquad j = i, \ldots i$$

where AOP_j represents one the output values.

$$\text{if } V_i > AOMAX_i \qquad \text{or } V_i < AOMIN_i$$

for every i = 1, ..., 10 then the test fails, otherwise it is passed. One sample is taken on each iteration of the microprocessor program. The flow chart of the Analog Output Accuracy is shown in Figure 10.

J. Analog Input Accuracy

When a normal analog input (AI) channel is read the analog input specified by the Interface input/output Bus address is switched to the Analog-to-Digital Converter (ADC) module where it is converted to a digital signal and returned to the microprocessor. When the analog input accuracy check is exercised the microprocessor sets control bits of the address word and forces an output data word to the Analog-to-Digital (A/D) converter module. This data word is latched by using the Data Available (DA) strobe and is converted to an analog signal via a D/A converter. The "test" value can then be read back via the Data Request (DR) strobe. The output test value is compared to the input value and if the ADC is outside a given tolerance, an error is declared as shown in Figure 11 [8].

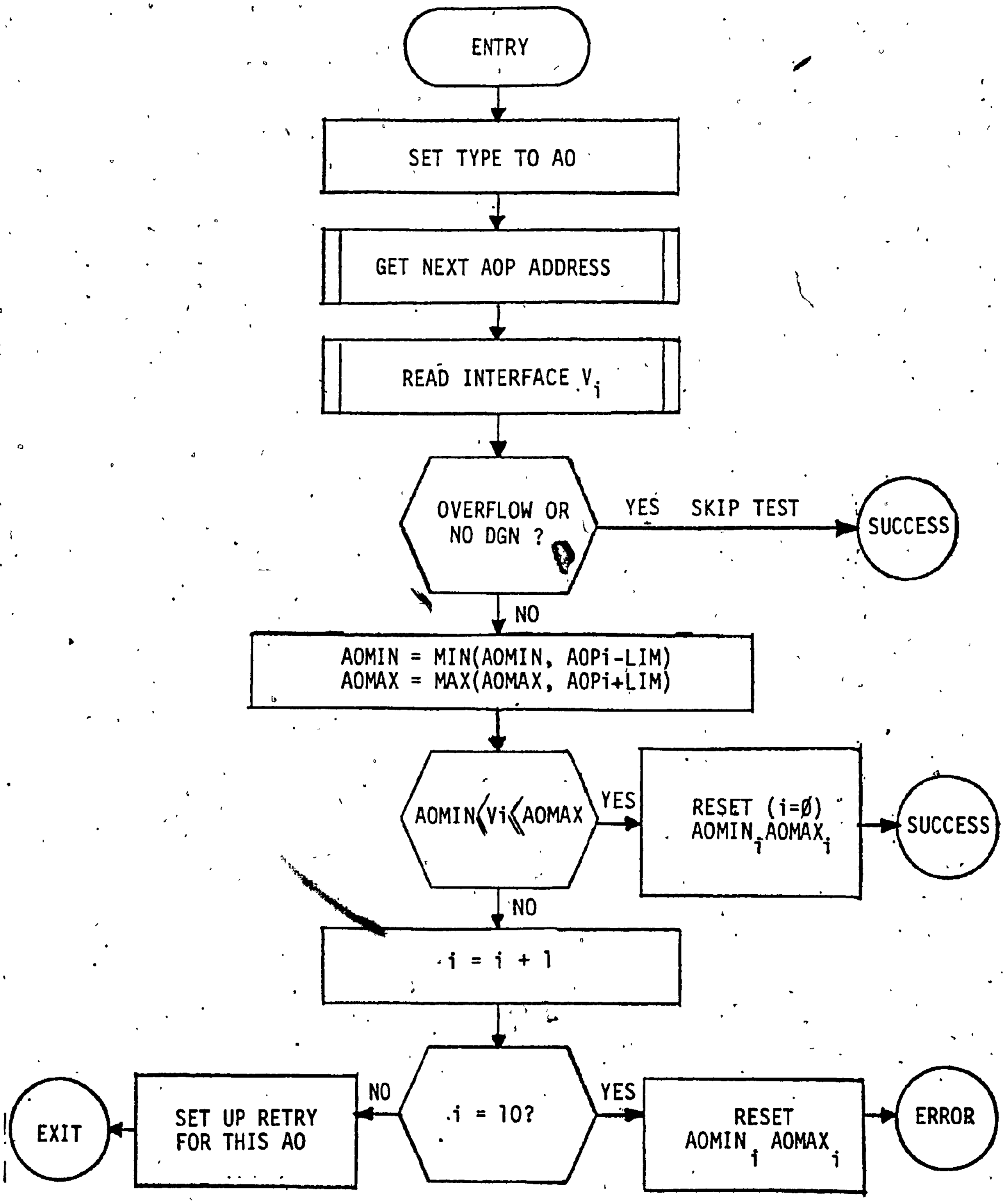


Figure 10. Analog Output (AO) Accuracy Flow Chart.

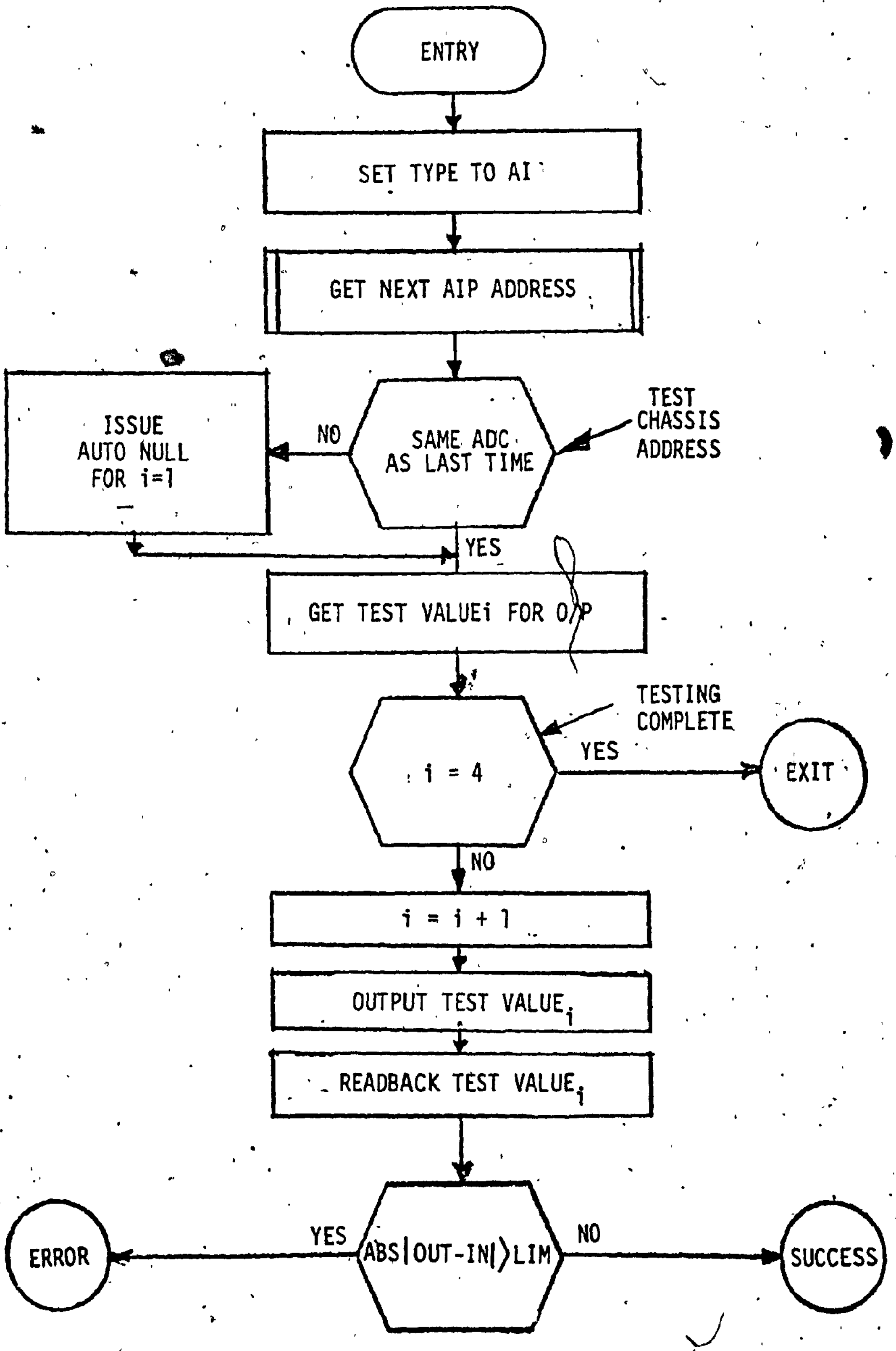


Figure 11. Analog Input (AI) Accuracy Flow Chart

V. 4-BIT-SLICE MICROPROCESSOR APPLICATION IN FLIGHT SIMULATION

A. General

This emulator is a 16-bit microprocessor system used in flight simulator applications (Figure 12) in conjunction with the main processor which is a minicomputer system. The microprocessor controls and processes the data flow between the computer system and the flight simulator compartment. In addition it relieves the main computer by executing function generation, special requests and diagnostic routines [2].

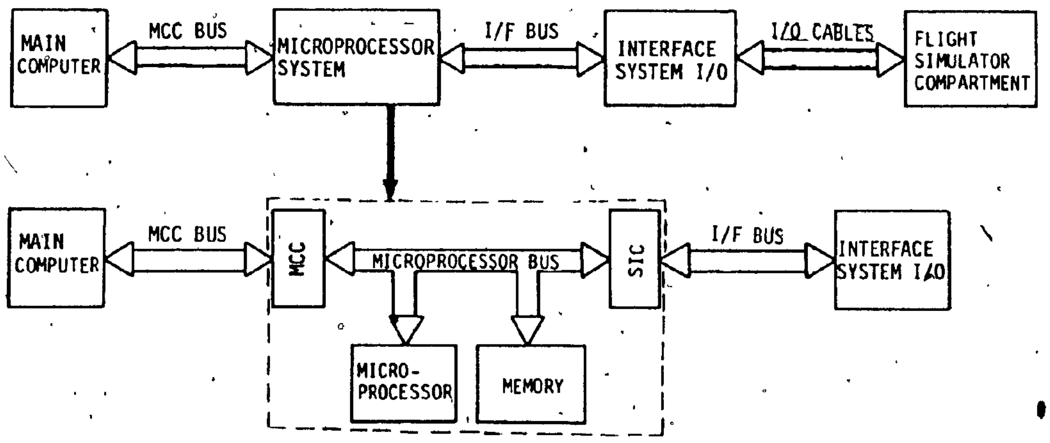


Figure 12. Computer Flight Simulator System Block Diagram

The microprocessor system is actually a computer Interface Unit (CIU) that handles the communication between the main computer and the Input/Output Interface System [2]. The microprocessor contains four modules which handle two distinct functions. The first is the controlling function exercised by the microprocessor module in conjunction with the memory module.

The second is the interfacing function done by the Main Computer Controller (MCC) and the Simulator Interface Controller (SIC) as shown in Figure 12.

B. Microprocessor Module

The microprocessor module is the heart of the emulator system incorporating all the control, computation, processing and decision making capabilities. The microprocessor effectively performs the arithmetic and logic operations as well as controls the address and data busses. It also provides the input/output strobes needed for the interfacing function that the Main Computer Interface Controller (MCC) and the Simulator Interface Controller (SIC) modules perform as shown in Figure 13. The microprocessor fetches instructions (macroinstructions) from the memory module via the microprocessor bus which is an internal bus. Each instruction consists of several microinstructions which are executed in sequence. All microinstructions are microcoded, and each one consists of 48 control signals contained (stored) in the 48-bit pipeline register [8]. The Microprocessor consists of the following units:

a. Computer Control Unit (CCU)
b. Arithmetic Logic Unit (ALU)
c. Program Control Unit (PCU)
d. Bus Source/Destination Control (BC)
e. Input/Output Control (IOC)
f. Interrupt Control (INTC)

The microprocessor is a Central Processing Unit (CPU) multi-chip implementation. The CPU is divided into two major sections, the Computer Control Unit (CCU) and the Processing Unit, each with distinct multi-functional

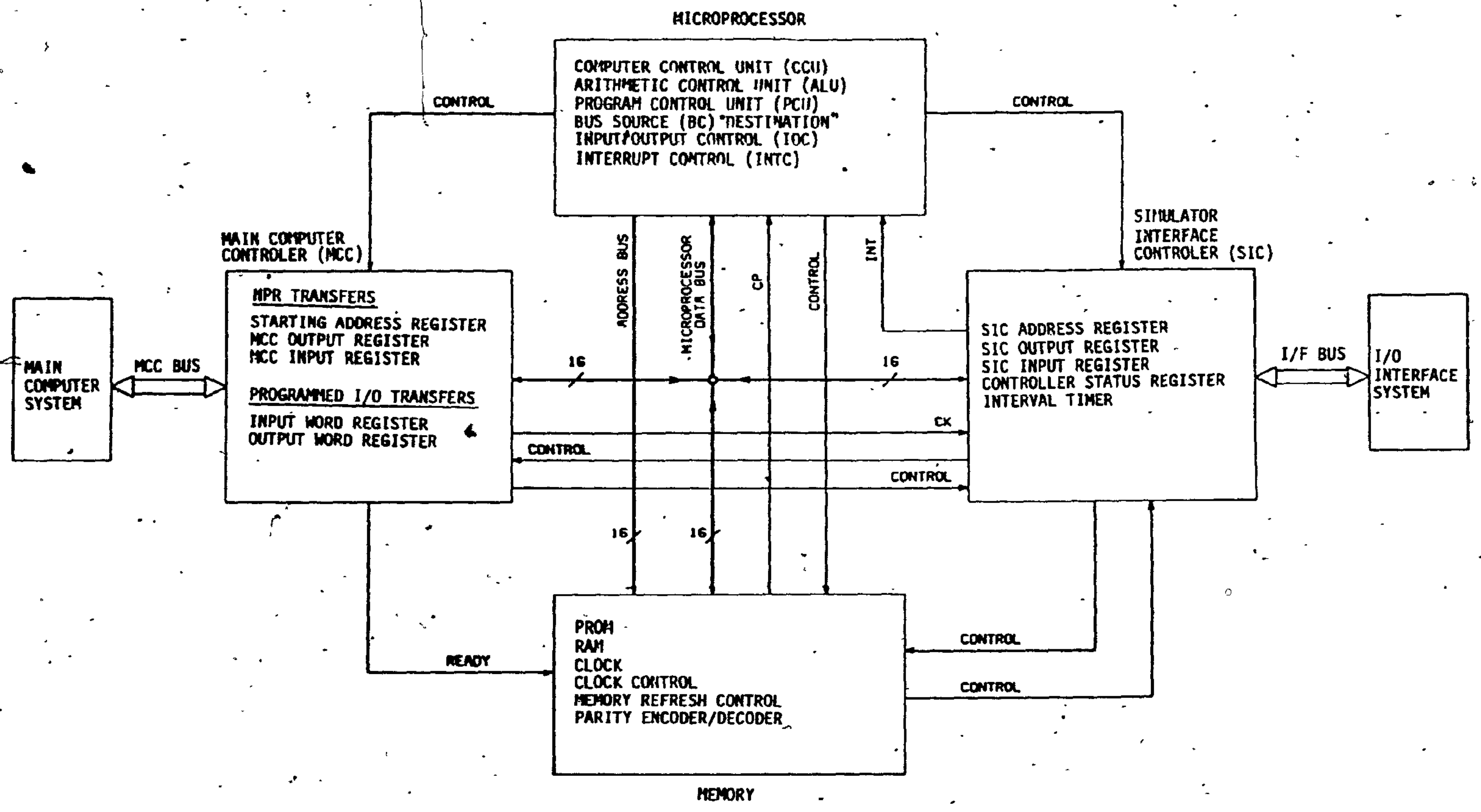


Figure 13. Microprocessor Computer Interface System Block Diagram

operation. Additionally the Input/Output Control (IOC) and the Interrupt Control (INTC) which are controlled by the microprocessor can be treated as separate functional sections (Figure 13). The microprocessor module operates on a 16-bit address bus and a 16-bit data bus. The program routine, once fetched from the memory module is fed to the computer control unit which instructs the arithmetic logic unit how to process the data. The computer control unit also drives the input/output control unit which selects the source and the destination of the data transfer [3].

Computer Control Unit (CCU) - (Figure 14)

The computer control unit consists of the following units:

(1) Pipeline Register (P/L): The pipeline register is a 48-bit parallel output data register. The register content controls the input/output data selection from the main computer controller and the simulator interface controller modules as well as the interrupt mode of communication. The pipeline register data also drives the program control unit (PCU), the Shift Control (SC) and the Carry-In-Control (CIN). It also controls the Microsequencer Counter, the Condition Code Multiplexer (CCM), the Source (SR), the Function (FN) and the Destination (DS) modifier.

(2) Instruction Register (IR): The instruction register is a 16-bit data register loaded by the memory module. The eight most significant bits, which are fed to the PROM map, are Operating Codes (OP-CODES), for the CCU. The eight least significant bits are two 4-bit ALU register designations which are multiplexed by the A, B, modifier, under the control of pipeline register output signals.

(3) PROM Map: The Prom Map is addressed by 8-bit instruction register operating code, and enabled by the microsequencer. The addressed memory location contains an 8-bit word that sets the stack pointer in the microsequencer to the appropriate location.

(4) Condition Code Multiplexer (CCM): The condition code multiplexer multiplexes the arithmetic logic unit output flags from the program status register to modify the counting steps of the microsequencer. The multiplexing function is controlled by P/L register output Condition Code (CC) and True/False (T/F).

(5) Microsequencer: The microsequencer initializes the P/L register in the initial start up procedure. After the startup, the microsequencer stack pointer location is determined by the PROM map output. Having set the stack pointer, the microsequencer counts a sequence of numbers which corresponds to memory location in the PROM micromemory. At each count, a memory location in the PROM memory is addressed, and the P/L register reads and stores the contents of the addressed location. The microsequencer count is modified by the condition code multiplexer and P/L register output Next Address Control (NAC). The modification is performed during the execution of the programmed routine. When the PROM map is disabled for read operation the microsequencer stack pointer can also be set by the P/L register output Jump Address (JA).

(6) PROM Micromemory: The PROM micromemory contains the following microprocessor routines:

a. Arithmetic functions
b. Logic functions
c. Program control (looping, jumping, branching)
d. Memory read/write functions
e. Input/output interfacing functions
f. Internal ALU register control
g. P/L register data flow

(7) Program Status Register (PSR): The program status register is a 4-bit data register that stores the condition of the four ALU output flags. The loading of the register is regulated by the input/output unit.

(8) Program Control Unit (PCU): The program control unit steps the ALU through the program instructions and sequences all RAM and PROM address in the memory/clock module. The PCU consists of a sequence (counter) which is set and driven by the P/L register.

Processing Unit - (Figure 14)

The processing unit consists of the following units:

(1) Data Register: The processing unit data register stores 16-bit input from the data bus. The data register is clocked by P/L register output signal. The output of the data register can go either to the ALU or to a 4 to 16 decoder or to a 16 to 4 encoder.

(2) 4 to 16 Decoder: The 4 Least Significant Bits (LSB) of the data register can be used as a condition code for the CCU. Because of Z however, the 4 LSB can also go to a 4 to 16 decoder to be used to build a mask and to set, reset or compare a specific bit in an ALU register.

(3) 16 to 4 Encoder: The output of the encoder is a 4 bit hexadecimal number (0-F) that indicates the first most significant zero bit in the word located in the data register.

(4) A-B Modifier: The A-B modifier is controlled by P/L register output to multiplex two 4-bit inputs from the CCU instruction register. The multiplexed output specifies the selection of two internal ALU registers used during the operation cycle.

(5) Source (SR), Function (FN), Destination (DS), Modifier: The SR, FN, DS modifier multiplexes P/L register output signals to produce control bits used for the operation of the ALU unit.

(6) Arithmetic Logic Unit (ALU): The ALU consists of four 4-bit slice processors that function in a parallel mode. It is clocked by the memory module and operates on 16-bit input data from the data register. The ALU functions and operations are controlled by the following units [1]:

a. A-B modifier
b. Carry-In Control (CIN)
c. Shift Control
d. SR, FN, DS modifier
e. Program Control Unit (PCU)
f. Pipeline Register (P/L)

The data read from memory is loaded into the data register. The output of data register can go to a 16-to-4 ENCODER, a 4-to-16 DECODER or to the ALU. The output of the ALU (Y) can be put on the data bus. The operation performed by the ALU depends on 9 signals coming from the pipeline (SR, FN, DS). The ALU result produces 4 flags (Z, V, N, C) (Zero, Overflow, Negative, Carry), which can be loaded into the Program Status Register (PSR).

(7) Shift Control and Carry Control (CRC): The shift control determines what will be the new bit (LSB or MSB) during a shift operation, the shifted out bit will be the carry (C) bit into the PSR. The "carry in" of the least significant slice comes from the Carry In Control (CIN), the other "carry in" comes from a Look Ahead Carry (LAC). The ALU has 16 internal registers. The source register address (A) and the destination register address (B) come from the A-B modifier in order that, under the pipeline register control, A and B can be any of RS, RD from the instruction register or X1, X2 from the pipeline.

Input/Output Control (IOC) Unit - (Figure 14)

This unit controls which input/output registers in the simulator interface controller or in the main computer controller will be the source/destination to the microprocessor data bus. During an input/output instruction, one strobe will be issued depending on the output/input control and the 4 least significant bits of the OP-CODE. The destination of the microprocessor data bus is controlled by 5 dedicated P/L bits. Any, none or all of them can be set to indicate the loading of data into the designated register.

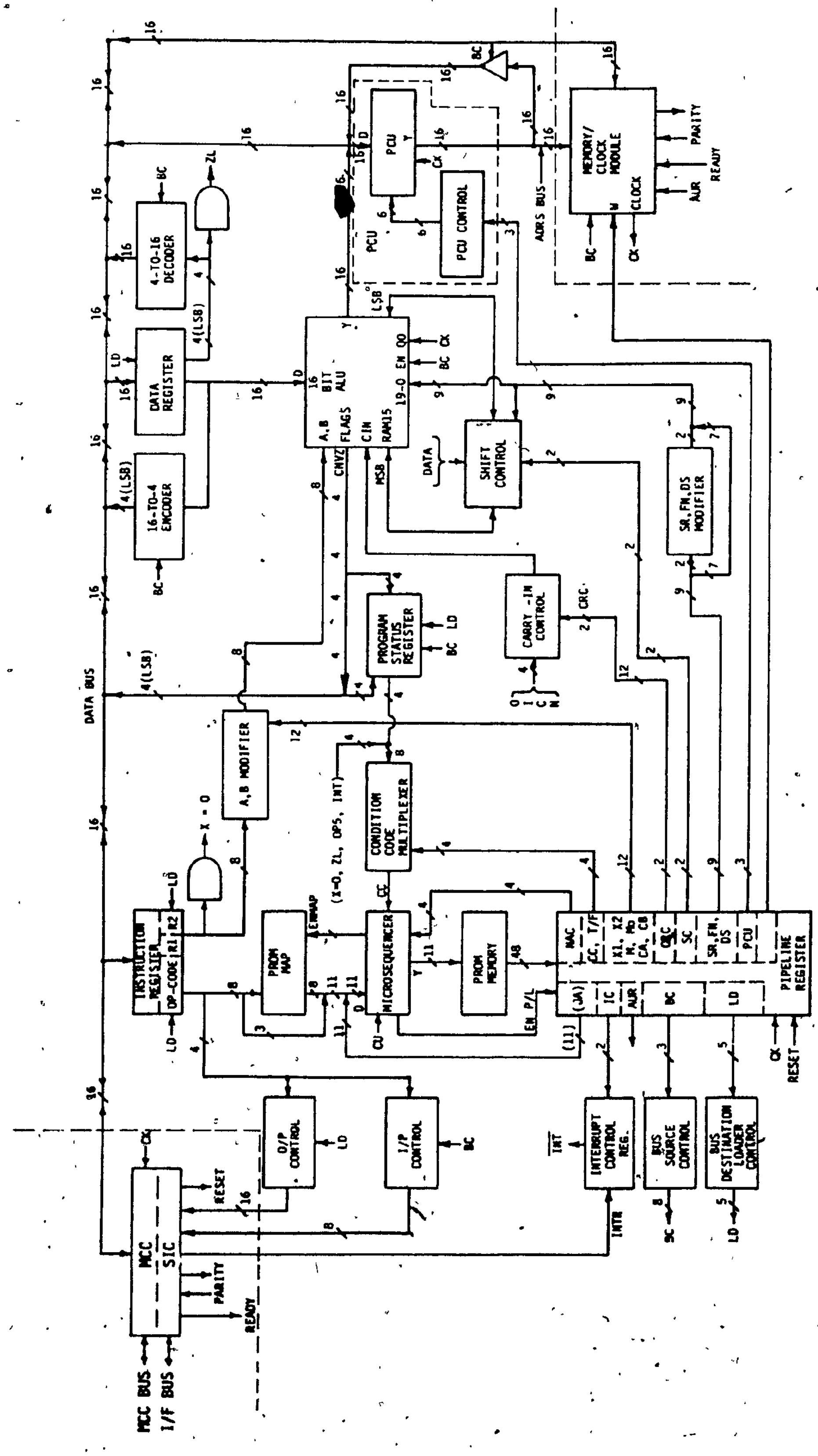


Figure 14. Microprocessor Module Block Diagram

Interrupt Control (INTC) Unit - (Figure 14)

This unit handles the micro-control of interrupts. This is done by the P/L control bits Jump Address (JA), Interrupt Control (IC) which control the status of the interrupt control register. The Enable Interrupt (EI) macroinstruction allows all the interrupts to interrupt the microprocessor. This interrupt, however, originates in the simulator interface controller status register and whenever it is enabled causes the computer control unit to jump into the interrupt service routine at the next instruction fetch [8].

C. Memory Module

The memory module contains a phase timing cycle generator and PROM (Programmable Read Only Memory) and RAM (Random Access Memory) components which supplement the operation of the microprocessor module. The PROM memory stores the microprocessor system routines and functions. The RAM memory is used by the microprocessor for temporary data storage and retrieval. The Memory module contains the following units:

a. Programmable Read Only Memory (PROM)
b. Random Access Memory (RAM)
c. Clock
d. Clock Control
e. Memory Refresh Control
f. Parity Encoder/Decoder

The memory module functions are controlled by microprocessor output control signals, Memory Write (MW) and Memory Read (MR-). These two signals are processed by the memory control logic to generate a multi-phase timing cycle which will determine a read or write operation, and generate multiplexing signals that control the addresseing of RAM components. In either read or write operations, a 16-bit address from the address bus is stored in the memory address register. The eleven least significant bits of the input address are memory locations. The remaining five bits determine RAM or PROM ~~addressing.~~

Data is transferred into and out of the memory module via a 16-bit bi-directional data bus. The tristate data buffer, which is controlled by the memory control logic, routes the flow of data between the data bus and the memory module. The memory module block diagram of operation is shown in Figure 15 [3]. The control signals involved in the operation of the memory module are;

a. Memory Read (MR-) Indicates a read function
b. Memory Write (MW) Allows the RAM memory to store data
c. Row Address (RAS) Indicates row addressing of RAM memory
d. Column Address Strobe (CASS) Indicates RAM column addressing
e. Refresh Address (RFADRS) and Column Address (CADRS) Multiplexe input address for the memory address register to generate a 7-bit address
f. Refresh Row Address (RFRAS) Refreshes the Ram memory contents during RAM addressing.

Clock and Clock Control

The clock is a multiphase free running clock used to control the operation of the memory and the microprocessor module.

Memory Address Register

The memory address register is clocked by the memory control logic to store a 16 bit address from the address bus. The three most significant bits specigy PROM memory addressing and feed into the PROM memory block decoder. The next two significant bits indicate RAM memory addressing and feed into the RAM memory block decoder. The eleven least significant address bits comprise the actual RAM or PROM address location.

Tristate Control Buffer

Through the tristate buffer, the RAM memory block decoder controls the loading and column addressing of RAM memory components. Input signal CAS indicates RAM column addressing, and input signal WRITE indicates the loading of input data from the tristate data buffer into the RAM memory.

RAM Memory Multiplexer

Through the RAM memory multiplexer, memory control logic output signals RFADRS and CADRS multiplex the 14 least significant bits from the memory address register. The multiplexer generates a 7-bit output which address the RAM memory components.

Tristate Data Buffer

The tristate data buffer controls the routine of data to the data bus during read/write operations. During a read command from the microprocessor module, memory control logic output MR- allows the tristate data buffer to route data from the PROM or RAM memory to the data bus. In a write command from the microprocessor module, output signal MR- enables the tristate data buffer to route 16-bit input from the data bus to the RAM memory.

PROM Memory

The PROM memory is preprogrammed with the microprocessor operating procedures and subroutines. The memory locations are addressed by the 11 least significant bits in the memory address register. The PROM is enabled from a read function by the PROM memory block decoder and the memory control logic. The memory contents, during a read command, are loaded on the data bus through the tristate data buffer.

RAM Memory

The RAM memory stores input data, from the 16-bit data bus, which can be retrieved by the microprocessor module during the execution of subroutines. The data is transferred between the RAM memory and the data bus through the tristate control buffer. The RAM memory is addressed in two stages. A row of RAM components is addressed by the RAM memory multiplexer when the RAS- signal is active. After row addressing, output signal CAS- enables a column of RAM components to be addressed by the RAM memory multiplexer. Data is loaded into the RAM by a WRITE command after the completion of addressing.

Memory Read, Write and Refresh Operations

This subsystem is controlled by Memory Read (MR-) and Memory Write (MW) microprocessor signals and three functions can be activated. Memory Read occurs when MR- is active (MW inactive) and can be RAM or PROM read operation depending on the memory block address decoder (selection). Memory Write occurs when MW is active (MR- inactive) and deals with the RAM memory. The third function is the Read/Refresh (Read

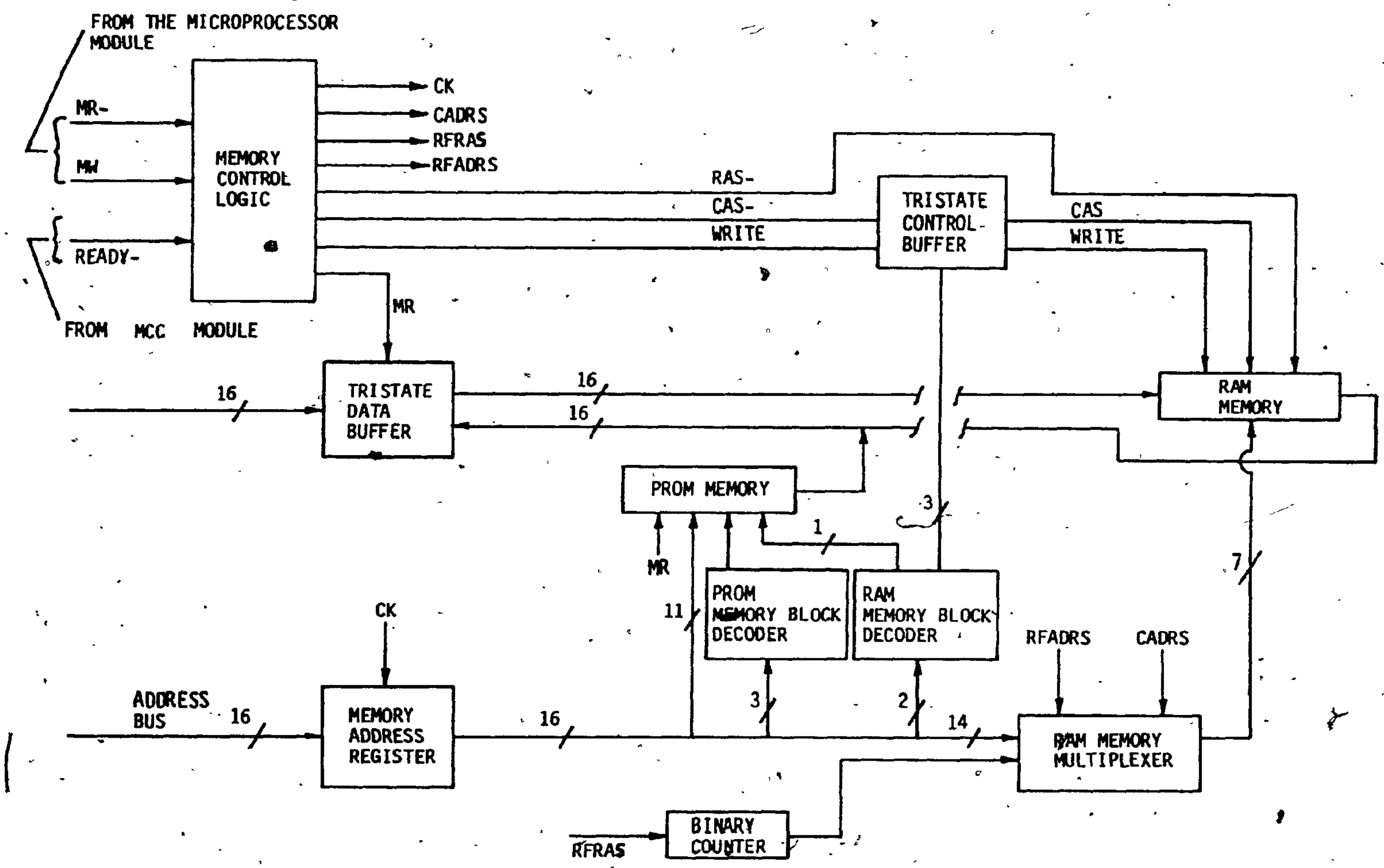


Figure 15. Memory Module Block Diagram

followed by Refresh) operation, which occurs when MW and MR- are both active. Finally when MW and MR- are both inactive no memory operation is performed at all [2].

D. Main Computer Controller (MCC) Module

The main computer controller module performs bus interfacing address and data buffering, address incrementing and decoding, transfer error detection, and bus controlling function between the main computer bus and the microprocessor system as shown in Figure 16. When the microprocessor is ready to process data from the main computer bus it signals the main computer controller module to store the input data in the main computer controller input/output registers. The BUSY/READY signal initializes the memory module to provide the microprocessor with a routine through the data bus. While executing the procedure, the microprocessor accessed the RAM and PROM components of the memory module through the address bus for processing and decision making. The final data is stored either in the simulator interface controller module input/output registers or in the memory module RAM components for future use. The microprocessor selects a memory location (RAM or PROM) in the memory module via the address bus. All other data transfers, between any two modules are channeled through the data bus. There are actually two types of transfers that the main computer controller module performs, Non-Processor Request (NPR) and Programmed Input/Output (PIO). During the NPR transfers which are also called Direct Memory Access (DMA), the microprocessor is the bus master and the main computer memory is the slave, while during PIO transfers the main computer is the bus master and the microprocessor is the slave [3]. The Main Computer Controller Module contains the following units:

a. Starting address register
b. Address decoder
c. Output data register
d. Input data register
e. Computer transfer controller
f. Interrupt controller

Through the address decoder and the main computer address bus, the computer system addresses the microprocessor to perform the interfacing routines. When the microprocessor is addressed, the address decoder generates a device select (DEVSEL) signal which enables the main bus transfer controller (Figure 16). The Main Computer Controller (MCC) module processes input/output control signals from the microprocessor module and the main computer bus. The main computer controller module performs input/output register selection, controls the accessing of the computer system memory for non-processor use, controls the input/output data transfer, and enables the interrupt controller for the interrupt mode of communication between the microprocessor through the input register. When data is available, the computer transfer controller which processes input/output control signals from the main computer bus and the microprocessor module, enables the loading of data into the input register. The microprocessor reads the register contents through the data bus.

Similarly, data is tranferred from the microprocessor to the main computer controller bus. When data is available on the microprocessor data bus, the microprocessor module writes the 16-bit data into

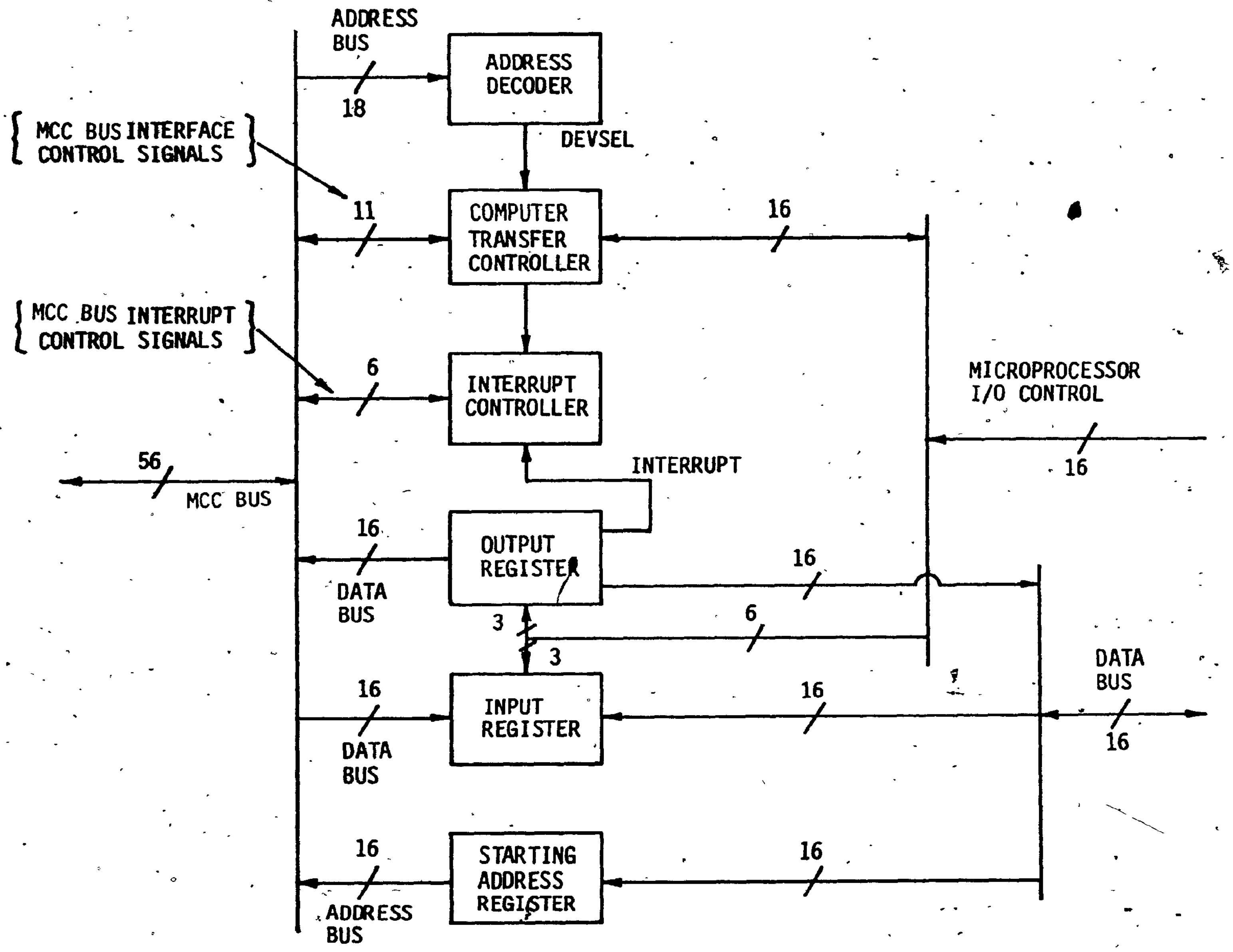


Figure 16. Main Computer Controller (MCC) Module Block Diagram

the output registers. The input/output control from the microprocessor module loads the output register with the input data. The computer system accesses the output register via the main computer controller bus and reads its contents (data lines DØØ through D15). The output register is enabled for accessing by the computer system through the computer transfer controller operating the computer system in the non-processor mode, the microprocessor module accesses the computer system memory through the main computer controller module data bus. The memory locations are addressed by the microprocessor through the main computer controller address bus and the starting memory address register [6].

E. Simulator Interface Controller (SIC) Module

The simulator interface controller module performs bus interfacing, address and data buffering, address incrementing and decoding, transfer error detection, and bus controlling function as it is shown in Figure 17. After the data from the interface bus has been processed the resulting output is stored either in the main computer controller input/output registers or in the memory module RAM elements. The data stored in the main computer controller registers can be accessed by the main computer system and used in the simulation program. The Simulator Interface Controller Module contains the following units:

a. Address register
b. Output register
c. Input register
d. Error controller and status register
e. Interface transfer controller
f. Interval timer

The Interface Transfer Controller processes input/output control signals from the interface bus and the microprocessor this enabling data transfers between the interface bus data lines and the microprocessor data bus. The data flow is buffered through data registers and differential components (Figure 17). When a device on the interface bus is addressed, output data from the data bus is stored in the address register. The input/output control lines from the microprocessor enable the differential transmitter to load the interface bus with the address through the Interface Transfer Controller. After addressing is complete, the microprocessor loads the output data register with the data for the addressed device. The data is loaded on the interface bus through the differential transmitter. Data from the interface bus is received by the microprocessor through the differential receiver and the input data register. The interface bus input/output control signals, which are processed through the Interface Transfer Controller, enable the input data register and the differential receiver to transfer the data. The error controller and status register monitor control signals from the interface bus and status signal from the main computer controller module. A 16-bit output word, which is processed by the microprocessor module, determines the condition of the data between the main computer controller bus and the interface bus [3]. The main errors detected by this process (General Transfer (GT) Error, Parity (P) Error, Address Time Out (ATO) Error etc.) were examined in Chapter IV.

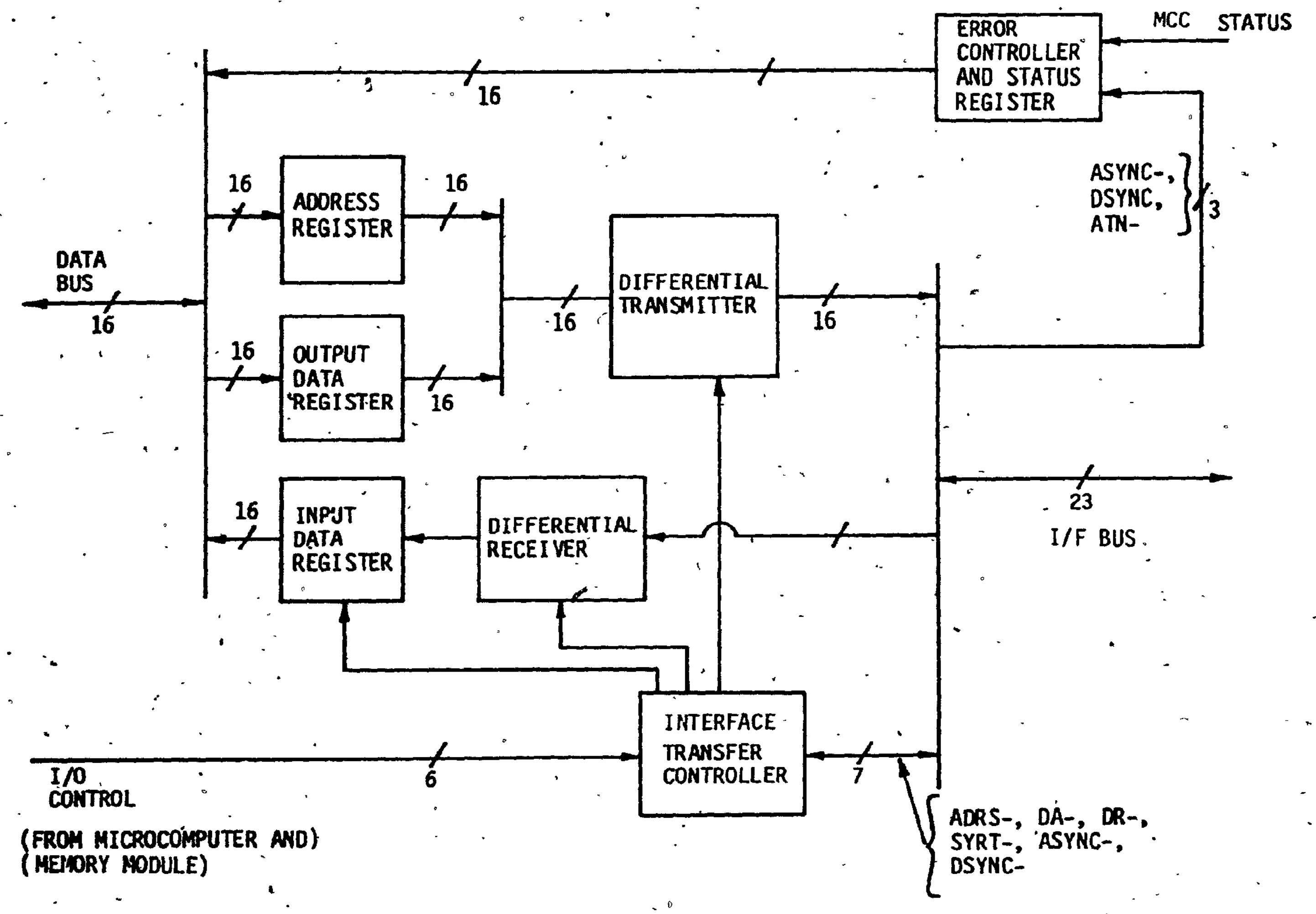


Figure 17. Simulator Interface Controller (SIC) Module Block Diagram

CONCLUSION

In this report the computer flight simulator sytems were examined and specially the microprocessor emulator unit. Particular attention was paid to the structure of the microprocessor and to the interaction between the microprocessor and the input/output interface system as well as between the microprocessor and the main computer system. In examining the interfacing techniques of the 4-bit-slice microprocessors we found that the emulator microprocessor provides the in-circuit-emulation facility as a complete system checking tool. The microprocessor system also provides increased speeds and flexibillity; and it is this property which makes the 4-bit-slice microprocessor feasible for computer flight simulation applications.

The 4-bit-slice microprocessors, however, have a disadvantage which is the system support software required in their applications. This results in increasing costs since highly specialized personnel have to be employed to meet the skills required in their application.

Finally, the overall application of the 4-bit-slice microprocessor in flight simulation involves transfers of data which are solely processed under the control of the microprocessor. Additionally, relief of the main computer system is achieved by the emulator which executes special functions on behalf of the main computer. The overall operation therefore, is greatly affected by the operation of the microprocessor since it must operate in a real time environment. This real time operation is accomplished by operating in a number of time frames and is controlled by a real time clock. Since a number of events (programs) must occur (run) within the time frame this might cause the operation to either run out of time or be very slow. These are the two critical limits under which the operation of the emulator microprocessor must function and which also govern the operation of the overall computer flight simulator system.

BIBLIOGRAPHY

1. Alexandridis, N. A., "Bit-Sliced Microprocessors Architecture", Microprocessors and Microcomputers, IEEE Computer Society, California, 1978, pp.69-92

2. Lesea, Austin and Rodnay Zaks, Microprocessor Interfacing Techniques, Sybex Inc., United States, 1978.

3. Artwick, Bruce A., Microcomputer Interfacing, Prentice-Hall Inc., New Jersey, 1980.

4. The AM2900 Family Data Book, Advanced Micro Devices, Inc., California, 1978.

5. Powers, Michael V., and Jose H. Hernandez, "Microprogram Assemblers for Bit-Slice Microprocessors", Microprocessors and Microcomputers, IEEE Computer Society, California, 1978, pp. 186-198.

6. Minicomputer Interfacing, Edited by Y. Parker, Miniconsult Ltd., London, 1975

7. Srini, Vason P., "Fault Diagnosis of Microprocessor Systems", Microprocessors and Microcomputers, IEEE Computer Society, California, 1978, pp. 286-291.

8. Leventhal, Lance A., Introduction to Microprocessors: Software, Hardware Programming, Prentice-Hall, Inc., New Jersey, 1978.