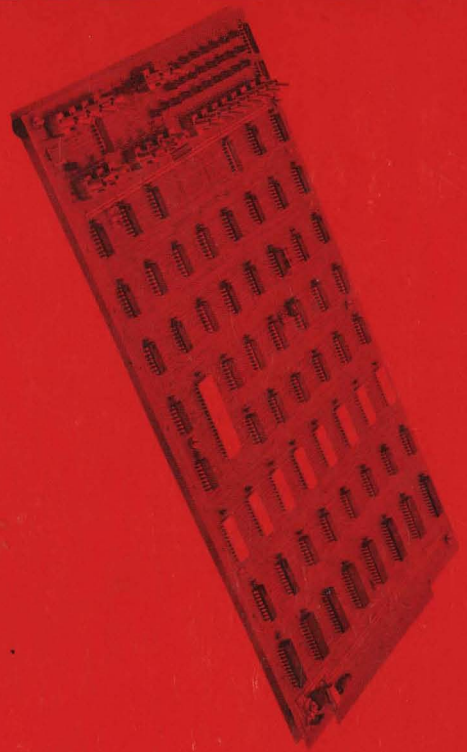
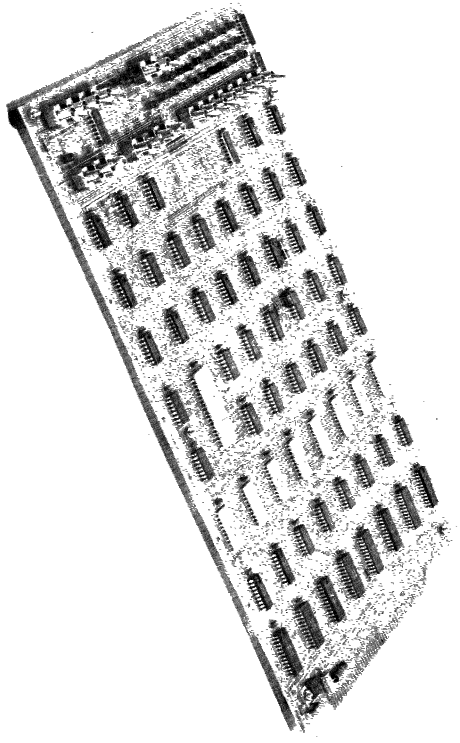
MACROLOGIC DEMONSTRATOR USER GUIDE



FAIRCHILD

MACROLOGIC DEMONSTRATOR USER GUIDE



FAIRCHILD

 • 464 ELLIS STREET • MOUNTAIN VIEW, CALIFORNIA 94042 • 415/962-5011 • TWX 910-379-6435

INTRODUCTION

Macrologic is a family of cost-effective LSI circuits designed for use in high-performance digital systems. Bit-slice design makes Macrologic components particularly suitable as functional building blocks for microprocessor-oriented applications, processor emulation and microprogrammed replacement of hard-wired logic. Microprogramming concepts give Macrologic a greater flexibility for many applications not found in fixed instruction-set microprocessors. As a bonus to the designer, both bipolar and CMOS Macrologic circuits are available. For faster system operation, high-speed bipolar Macrologic is the logical choice. Where power is at a premium, use CMOS.

The Macrologic Demonstrator is an educational tool designed to familiarize the user with the processor-oriented devices in the Macrologic family, and demonstrate concepts used in microprogramming.

The system consists of three main functional blocks. The 8-bit wide data path contains an ALU, a LIFO stack, a 16-register scratch pad and miscellaneous logic. The microprogram controller contains the control memory as well as microprogram addressing and sequencing logic. The operator control and display panel provides the human interface and allows the operator to load and modify the control memory, run or step through his microprogram, and monitor system status and contents of various system busses. The Demonstrator is completely self-contained. The only item that is needed externally is a +5 V power supply capable of delivering about 2 A of current. The Macrologic Demonstrator also has an S-100 bus interface which enables its control memory (RAM) to be loaded by an 8080 based system. Using the Demonstrator, a user can write a microprogram sequence, load and execute it and observe various states of the machine via LEDs.

TABLE OF CONTENTS

Section 1
BASIC THEORY OF MICROPROGRAMMING

The use of bit-slice bipolar Macrologic allows a machine to run at a higher speed than MOS microprocessors. Moreover, the modularity of Macrologic enables the data path width to be expandable in increments of four bits, allowing more flexibility than MOS microprocessors, whose data path width is fixed at 4, 8, 12 or 16 bits. However, unlike the fixed instruction set MOS microprocessors, Macrologic devices have to be microprogrammed. A prospective user has to become familiar with the concepts in microprogramming before he can utilize bit slices in his design effectively.

Basically, a microprogrammed machine is a 2-level machine. The machine command is known as the macrolevel. It takes a sequence of steps to execute a machine command; these steps are called microinstructions. In the case that the machine is a CPU, the basic machine command is the macroinstruction, which resides in main memory. After each macroinstruction is fetched, a corresponding sequence of microinstructions will be executed. Each microinstruction controls the machine for one clock cycle (called microcycle). A collection of microinstructions is called a microprogram. Microinstructions are stored in a memory called microprogram memory or control store. Usually PROMs are used for control store. However, the microprogram memory can be implemented using RAMs, which can be dynamically altered during machine execution; this is called writable control store.

A microprogrammed system generally consists of two sections *(Figure 1-1)*:

1. The controller section includes the microprogram memory and the sequencing logic required to provide the address to the microprogram memory.

2. The data path usually contains ALUs, general registers, stacks, etc. and is controlled by the appropriate bits fields of the microinstructions.

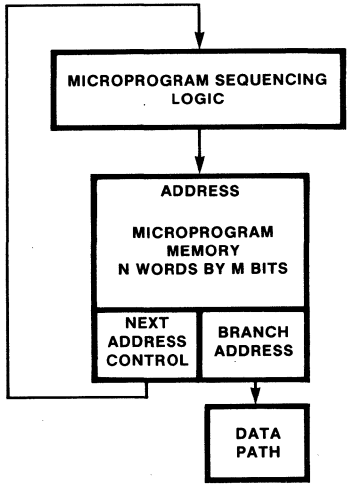


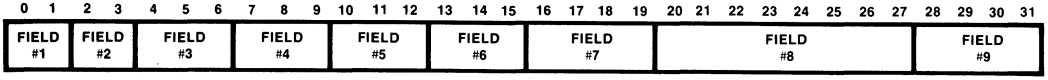
Fig. 1-1 A Generalized Microprogrammed System

The microprogram memory is simply an N-word by M-bit memory used to hold the various microinstructions. Each word of the microprogram memory contains M bits, which are usually grouped into various fields. The field can consist of various numbers of bits. The definition of the various fields of microprogram word is referred to as FORMATTING. *Figure 1-2* illustrates the format of a 32-bit microprogram word containing nine fields.

In general, a microinstruction must contain the following information: First, the definition and control of the operations to be performed by the data path and other system logic. These include such things as ALU function, ALU source, destination registers, shift or rotate control, stack operations, data in, data out select and so forth. Second, the address of the next microinstruction to be performed. In some cases, it can be implicit, such as when next address is current address plus one. In other cases, the actual value of the address field has to be supplied.

The use of microprogramming allows the designer to employ memory as one of the functional building blocks in logic design. Unlike hard-wired logic approach, where a designer generates timing and control signals by using combinations of gates and flip-flops and prays he does not have to make major changes in the future, microprogramming enables highly ordered and structured designs. Therefore, implementation of design changes is easier. Instead of tearing up gates and flip-flops, a change can usually be accomplished by reprogramming the control store. With memory prices dropping the way they are, the increased use of microprogramming in logic design is imminent. Microprogramming will be used in such applications as:

Special purpose controllers
High-speed (relative to MOS microprocessor) functions
Extending computer instruction sets
Emulations
Diagnosis/fault tolerance
Multiprocessors
Multi-level hierarchies of processors
Direct execution of high-level languages
I/O controllers



FIELD DEFINITION

FIELD 1 — DATA IN CONTROL
2 — DATA OUT CONTROL
3 — ALU SOURCE OPERAND CONTROL
4 — ALU FUNCTION CONTROL
5 — ALU DESTINATION CONTROL
6 — CARRY CONTROL
7 — MISC. CONTROL
8 — BRANCH ADDRESS
9 — NEXT ADDRESS CONTROL

Fig. 1-2 Definition of 32-Bit Microprogrammed Word Fields

Section 2
MICROPROGRAMMED SYSTEMS USING MACROLOGIC

The controller part of the microprogrammed system can be implemented using the 9408 Microprogram Sequencer. The 9408 is based upon a 10-bit Program Counter (PC) and addresses 1K pages of microprogram memory *(Figure 2-1)*. The basic instruction of the sequencer is FETCH, in which the program counter is incremented after each clock and, in effect, makes the next address the same as current address plus one. This type of addressing will cycle the microprogram memory sequentially, but will not be very adequate for most practical systems. The 9408 also has many provisions for branching.

The first type of branching is *unconditional branching*. Here a new address is placed on the 9408 input bus and loaded into the program counter. The new address can come from one of four different sources, selected by the BRV_0 to BRV_3 instructions.

The second type of branching is *conditional branching*. Here the 9408 performs a test on one of four available inputs. If test conditions are met, the program counter is loaded with the address at the input bus (BRANCH). If the conditions are not met, the program counter is incremented (FETCH).

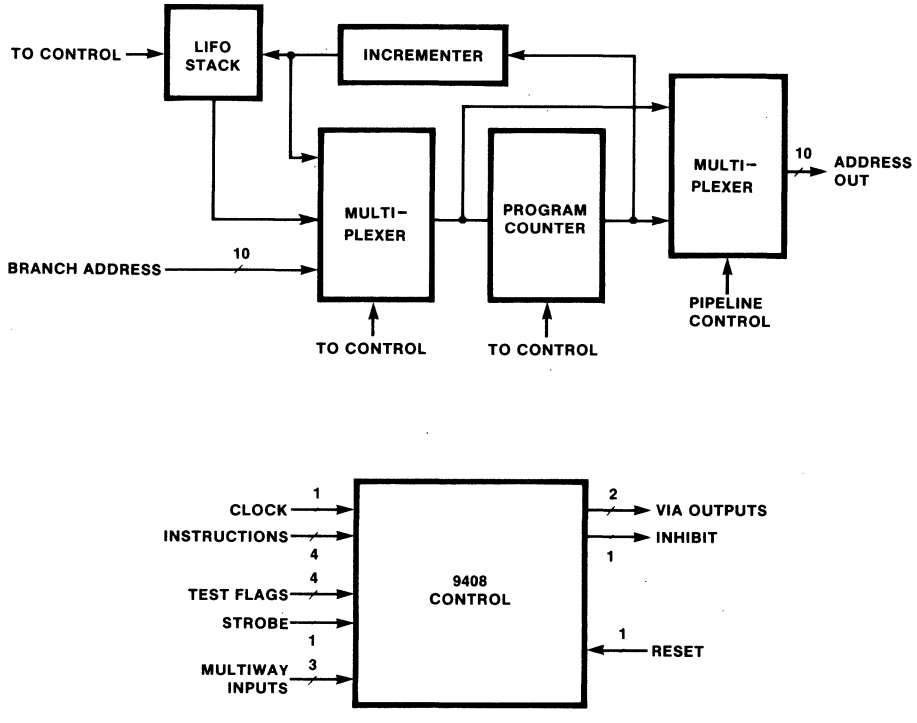


Fig. 2-1 9408 Microprogram Sequencer

The third type is the *multiway branch*. The multiway branch is similar to an unconditional branch in that a program counter will always be loaded into a specified address, i.e., a branch is certain to occur. However, the address of the branch is determined by substituting the three least significant bits of the branch address present at the input bus by the value of three multiway (test) inputs. The result is that the branch will be made to one of eight different locations.

The fourth type of branching is *subroutine branching*. In branching to a subroutine, the incremented value of the program counter is loaded into the on-chip LIFO stack. Then, the program counter is loaded with the address at the input bus. In returning from a subroutine, the program counter is loaded with the contents of the top level of the stack.

A microprogrammed system can be operated in either pipeline mode or non-pipeline mode. The non-pipeline controller is simply a 9408 and a control store *(Figure 2-2)*. In a pipeline system, an edge-triggered microinstruction register is also needed *(Figure 2-3)*. In a non-pipeline system, a microinstruction is read from the control store and executed before reading the next microinstruction. Since most microprogrammed systems are designed as synchronous machines, the maximum frequency the system will run at is indicated by the sum of the 9408 propagation delay (CP to address outputs), the memory access time, and the data path propagation delay.

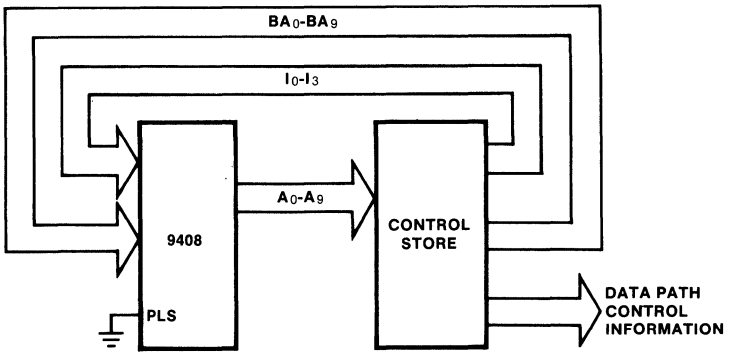


Fig. 2-2 Non-Pipeline System

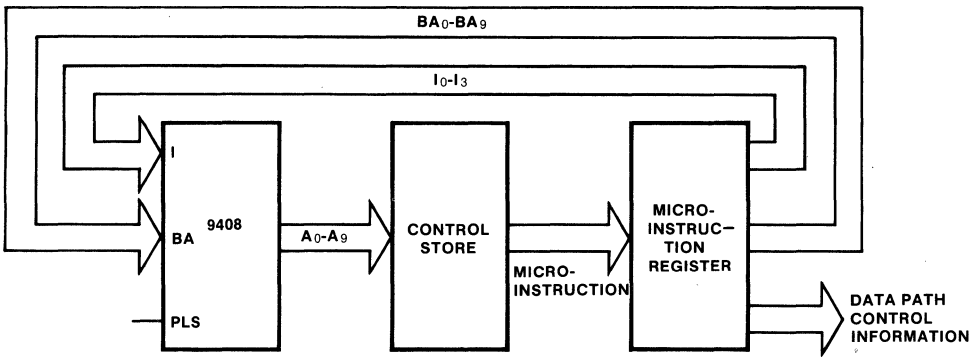


Fig. 2-3 Pipeline System

In the non-pipeline system, the data path logic is idle during the part of the microcycle when the control store is being accessed. Similarly, the memory is not cycled and is sitting with the same outputs during the part of the cycle when the data path is used. A more efficient use of the resources will be overlapping the execution of the current microinstructions with the fetching of the next microinstruction. This requires holding the current microinstruction in a microinstruction register *(Figure 2-3)*. Here, the control memory access time is hidden from the data path propagation delay. The system cycle time is the greater of the sum of the propagation, set-up, and access times of the controller or the data path worst case propagation delay. In many instances, a microprogram written for a non-pipeline system cannot be executed in pipeline mode. However, the 9408 architecture is chosen so that the same microprogram can be executed in pipeline or non-pipeline mode without any modification. In the Macrologic Demonstrator, the 9408 is designed to work in the non-pipeline mode.

The data path logic generally consists of ALUs, general registers, stacks, etc., and can be readily implemented with Macrologic devices (ALRS, DPS, etc.). To provide logical, arithmetic and storage capabilities, the 9405A can be used *(Figure 2-4)*. The 9405A is a 4-bit slice containing an eight-function ALU, eight general purpose registers, four conditional code flags (carry, negative, zero, overflow), an output register and the necessary decoding logic. It can be expanded in multiples of four bits without additional components, or can be expanded with a carry lookahead unit for greater speed.

Note the absence of a shifting network either before the RAM inputs or at the ALU outputs, which is quite common with other bit slice ALU designs. This is because Macrologic devices are designed to be functional building blocks in highly bus-organized systems so that sharing the input and output busses in the data path with the ALRS may be the hardware stack, general registers, etc. To eliminate the need for duplicating shifting capabilities in each Macrologic device, the shifting network is found in the 9404 Data Path Switch (DPS). The 9404 DPS is a combinational network consisting of two input busses and one output bus *(Figure 2-5)*. It performs functions such as arithmetic or logic shifting, masking, sign extension and constant generations. In addition, when connected to other Macrologic devices (for example, the ALRS), it allows a closed-loop system to be formed, so that the output bus can communicate to the input bus. *Figure 2-6* depicts a simple closed-loop system in which the ALRS inputs and outputs are connected to the DPS outputs and inputs respectively. External data may be introduced into the system using the K inputs of the DPS.

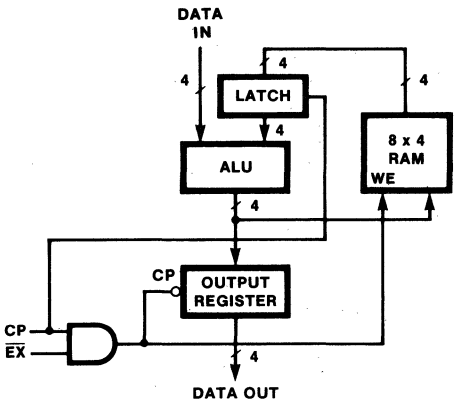


Fig. 2-4 9405A ALRS

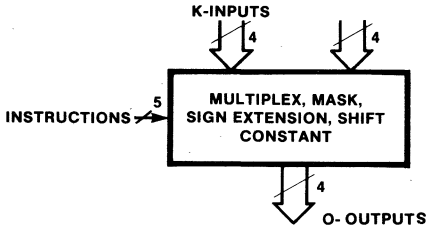


Fig. 2-5 9404 DPS

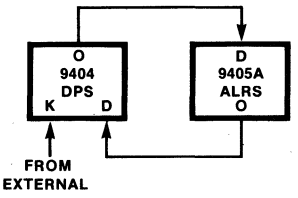


Fig. 2-6 A Simple Closed-Loop Data Path

As Macrologic devices are designed to be used in bus-organized systems, all devices are provided with 3-state data outputs and an Output Enable ($\overline{EO}$) input to control them. Therefore, the data outputs from the devices can be bussed together to obtain the output bus (D-bus) as in *Figure 2-7*. With a LOW level on the appropriate ($\overline{EO}$) input, a Macrologic device can be made to source data onto the output bus. The data inputs of the Macrologic devices can also be bussed together to obtain the input bus. Except for the DPS, Macrologic devices used in the data path are provided with individual execute ($\overline{EX}$) input and a device will not respond to the clock unless its $\overline{EX}$ input is LOW. Thus, the instruction inputs can be bussed together to obtain an instruction bus. The result is that during a single clock cycle, information can be routed from any device in the data path to any other device (including itself) by controlling the proper $\overline{EX}$ and $\overline{EO}$ inputs.

For some applications, a DPS and ALRS combination in the data path provides enough storage and computing power. For example, in an emulation design, one of the eight registers in the ALRS can be treated as the program counter, four can be accumulator registers, and the other three can be scratchpads. In some applications, this may be inadequate. If, for example, the emulation is for a machine with macrosubroutine nesting capabilities, the 9406 program stack can be added into the data path *(Figure 2-8)*. The 9406 is a 16-level, last-in first-out hardware stack. The top of the stack can be used as the program counter and the other levels provide return address storage for nested subroutines. In addition, it provides two status outputs to indicate Stack Full ($\overline{SF}$) and Stack Empty ($\overline{SE}$) conditions.

But suppose someone needed more than 16 levels of stack, say for unlimited levels of subroutine nesting capability. In that case, fixed level hardware stack will not meet the requirement. The approach may be changed to put the stack out in RAM (Random Access Memory) and maintain it by pointers. The 9407 Data Access Register can be used for this purpose *(Figure 2-9)*. Basically, the 9407 contains three registers with an adder network, an output register and a two-to-one multiplexer to select either the adder output or the individual register output. It is designed to perform memory address functions for RAM resident stack type applications. For example, the three registers can be used as program counter, stack pointers and operand address. The 9407 implements 16 instructions, which allows pre- or post-increment/decrement and register-to-register transfer in a single clock cycle. For applications which require more storage capacity than the eight registers the 9405A provides, the 9410 register stack (R-stack) can be added to the data path. The 9410 is a high speed 16-word by 4-bit memory with an edge-triggered output register *(Figure 2-10)*.

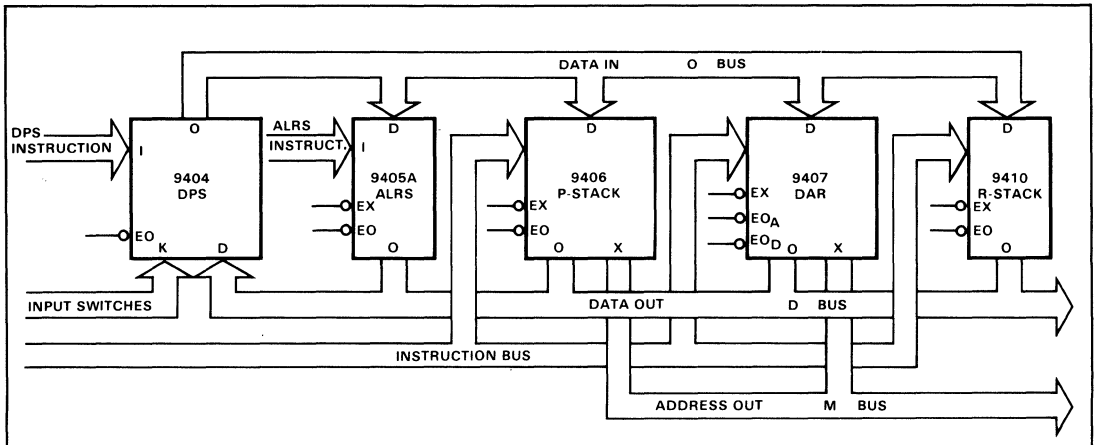


Fig. 2-7 Bussing Macrologic Arrays

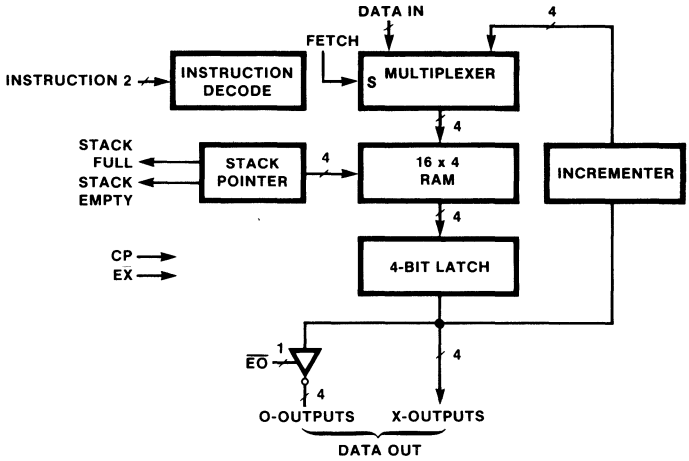


Fig. 2-8 9406 P-Stack

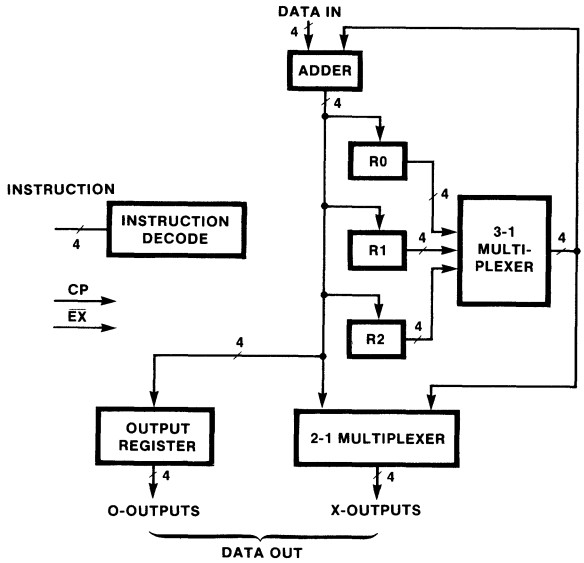


Fig. 2-9 9407 DAR

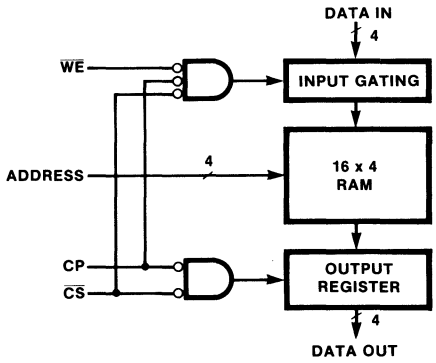


Fig. 2-10 9410 R-Stack

A SIMPLE PROCESSOR EXAMPLE

One of the many possible Macrologic applications is to implement emulators for existing instruction sets. These complex functional LSIs offer improved cost and performance while retaining software compatibility with the target machine. A simple 16-bit processor is a good example to demonstrate the ease of use and versatility of Macrologic.

The 16-bit fixed word-length processor, with four accumulators (AC_0-AC_3) and 2's complement arithmetic, has a 16-word push/pop stack for subroutine nesting, as well as general use. The memory reference instruction format is shown in *Figure 2-11*. The 2-bit index field in the instruction specifies four addressing modes — base page, PC relative, AC_2 and AC_3 relative. For the base-page mode, the 8-bit displacement field of the instruction is taken as the absolute address, i.e., first 256 memory locations. For the relative mode, the 8-bit displacement is treated as a signed number in 2's complement notation and added to the Program Counter (PC relative) or one of the specified accumulators (AC_2 or AC_3 relative). The result is then used as the effective address for the operand.

A data path suitable for this processor is shown in *Figure 2-12*. It consists of a 16-bit ALRS array, 16-bit P-stack array and 16-bit DPS array. The ALRS and DPS can perform all the arithmetic logic operations needed. The P-stack provides the required 16-level stack function. The ALRS has eight built-in accumulators, but only four are needed for this processor. The P-stack has the necessary features to implement the PC; however, if this feature is used, only 15 levels of nesting remain. This processor requires 16. Because the ALRS has four spare accumulators, one of these can be used as the PC, leaving three spares. Thus the PC feature of the P-stack is not needed and the address outputs are not used. The storage in the ALRS is allocated as follows: $R_0 = AC_0$, $R_1 = AC_1$, $R_2 = AC_2$, $R_3 = AC_3$, $R_4 = PC$, $R_5 =$ TEMP 1, $R_6 =$ TEMP 2 and $R_7 =$ TEMP 3. An edge-triggered Memory Address Register (MAR) on the output bus is provided. Data from the memory is introduced into the data path using one of the input ports of the DPS array. Data to the memory is obtained directly from the output bus. An edge-triggered Instruction Register (IR) is also provided to hold the OP code bits and index bits of the macroinstruction.

Figure 2-13 illustrates the microprogram control section for the data path. This control is centered around the 9408 sequencer operating in the pipeline mode. The INH output of the 9408 is used to share the control fields. Thus, the source, destination, and ALRS/P-stack control fields provide the 10-bit address for branching when needed. A 6-bit DPS control field provides the instruction inputs for the DPS array while the 4-bit SEQ field provides the instruction inputs for the 9408. Other fields lumped as miscellaneous are used to control the memory, etc.

The source and destination fields are decoded to activate the $\overline{EO}$ and $\overline{EX}$ inputs *(see Figure 2-13)*. Note that the IR clock and MAR clock signals are generated by gating the system clock with the appropriate destination decoder outputs. The Branch Address inputs (BA_0-BA_9) are obtained from a

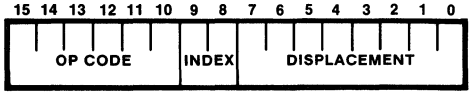


Fig. 2-11 Memory Reference Instruction Format

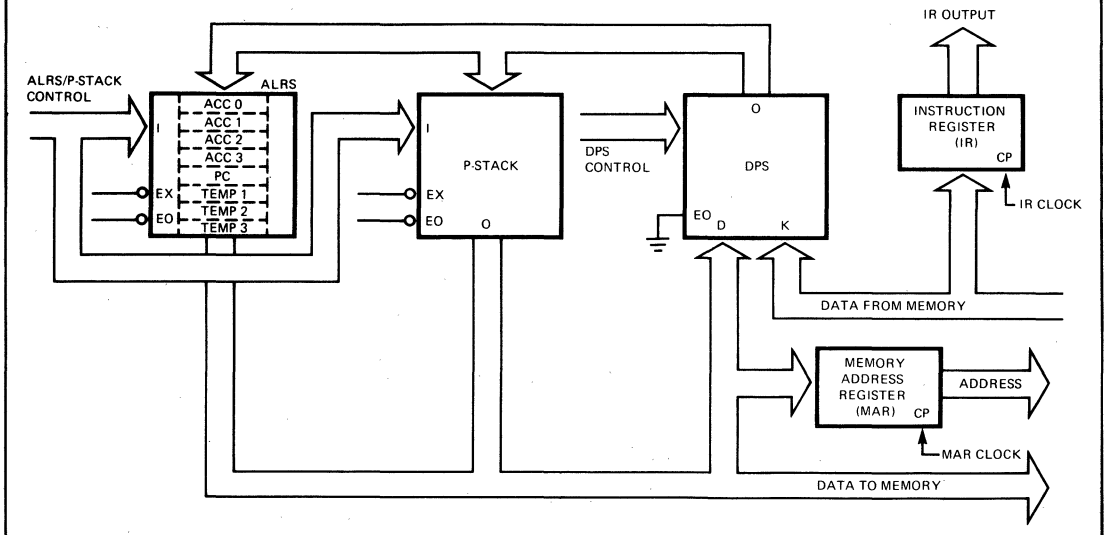


Fig. 2-12 Data Path for the Processor

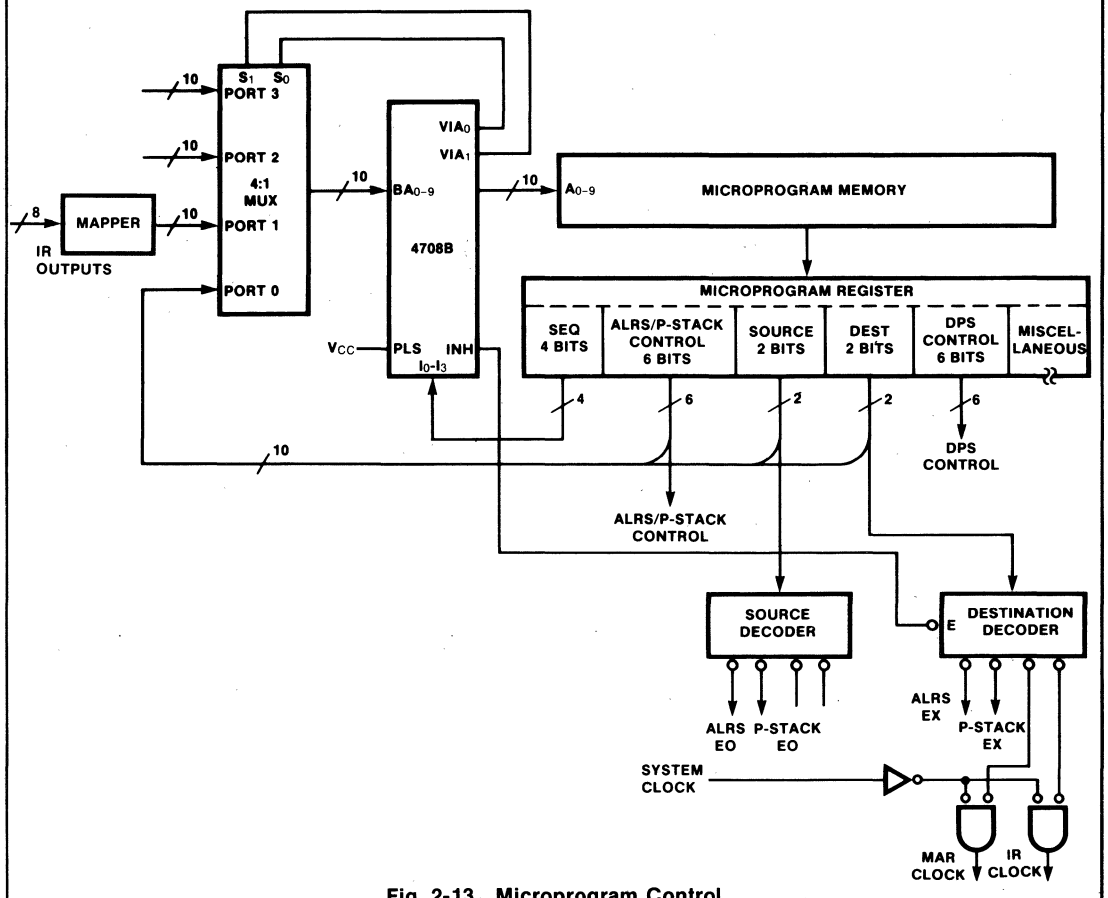


Fig. 2-13 Microprogram Control

4-way input multiplexer which, in turn, is controlled by the VIA_0 and VIA_1 outputs of the 9408. One of the inputs to this multiplexer consists of the address inputs for branching from the microinstruction register. The second part is fed by a mapper that may be a PROM or FPLA. It receives the IR outputs and translates them into a starting address in the control memory for emulation. *Figure 2-14* is a flow chart for the sequence of operations to accomplish macroinstruction fetch, while *Table 2-1* lists the operations performed by various data path elements and the 9408.

The first operation is to read the PC. Thus, the destination field specifies the ALRS and the destination decoder drives the $\overline{EX}$ input of the ALRS LOW. The ALRS/P-stack control field specifies "Read R_4." At the end of the microcycle, the contents of R_4, i.e., the PC, are in the output register of the ALRS. The SEQ field of the first microinstruction is FETCH; therefore, the 9408 generates the address of the second microinstruction.

Here, the ALRS is specified as the source and the MAR as the destination. The source decoder activates the $\overline{EO}$ input of the ALRS, the destination decoder enables the gating for the MAR clock, and the microinstruction loads the PC into the MAR. In the miscellaneous field, a memory read is initiated. The third microinstruction is made to increment R_4 by selecting ALRS as the destination specifying Add with Carry. The DPS outputs (ALRS inputs) are forced HIGH. This incrementation is in preparation for the next macroinstruction fetch. The result from the memory read operation, initiated during the second microinstruction, is now available on the K-bus of the DPS. The fourth microinstruction activates the IR clock so that the eight most significant bits of the memory data are loaded into the IR. Assuming the data is still on the bus, the sign extended displacement is loaded into R_5 (TEMP 1) of the ALRS in the fifth microcycle by selecting "Load R_5" as the ALRS operation, and selecting the "K-Bus Sign Extended" for the DPS. It should be recalled that the data path has 16-bit fixed word length and the displacement must be treated as a 2's complement number. By using the sign extension capabilities of the DPS, the sign bits, i.e., most significant bits, can be aligned. At this point, the instruction is in the IR and the sign extended displacement is in TEMP 1. The sign of the least significant eight bits of the macroinstruction is extended in anticipation of a memory reference instruction. The sixth microcycle is intended to decode the IR. By specifying a BRV_1 in the SEQ field, the VIA outputs of the 9408 select the mapper output as the source for next address. The mapper is designed to provide the starting address of the routine to emulate the instruction currently residing in the IR.

The total microprogram really consists of several simple routines. These easy steps can be converted into binary patterns to be loaded into the control store. Once a data path architecture and microinstruction format has been chosen for a given system design, the microprogram can be written to realize the desired function. It can then be assembled, using the microprogram assembler, to get the binary listing that specifies the control store address and contents. Using this information, the control store can be loaded with the program and the system is ready for operation.

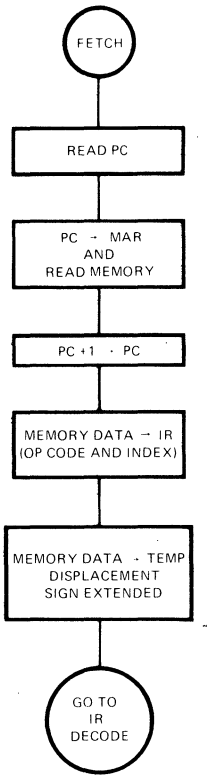


Fig. 2-14 Flow Chart For Fetch Operation

SOURCE	DESTINATION	ALRS/ P-STACK CONTROL	DPS CONTROL	SEQ	MISCELLANEOUS
DON'T CARE	ALRS	READ R_4 (PC)	DON'T CARE	FTCH	- - -
ALRS	MAR	DON'T CARE	DON'T CARE	FTCH	READ MEMORY
DON'T CARE	ALRS	ADD WITH CARRY TO R_4	ALL ZEROS	FTCH	- - -
DON'T CARE	IR	DON'T CARE	DON'T CARE	FTCH	- - -
DON'T CARE	ALRS	LOAD R_5 (TEMP 1)	K-BUS SIGN EXTEND	FTCH	- - -
DON'T CARE	DON'T CARE	DON'T CARE	DON'T CARE	BRV_1	- - -

Table 2-1 Operations for Fetch Instruction

Section 3
DESCRIPTION AND OPERATION OF THE MACROLOGIC DEMONSTRATOR

The Macrologic Demonstrator is an educational tool designed to familiarize the user with the processor-oriented devices in the Macrologic family. It consists of three main functional blocks *(Figure 3-1)*. The 8-bit wide data path contains an ALU, a register file, LIFO stack and miscellaneous logic. The microprogram controller contains the control memory, as well as microprogram addressing and sequencing logic. The operator panel provides the human interface and allows the operator to load and modify the control memory, run or single-step the microprogram and look at system status and contents of various system busses. An S-100 bus interface is also provided for optional automatic loading of the control store by 8080 based system.

Schematics for the Macrologic Demonstrator are located at the end of this section.

THE DATA PATH

Two 4-bit slices of the following Macrologic devices form the 8-bit wide data path *(Figure 3-2)*. The Data Path Switch (9404 DPS) allows shifting, constant generation and other operations. An Arithmetic Logic Register Stack (9405A ALRS) performs eight arithmetic and logic functions and includes eight data registers. The Program Stack (9406 P-stack) contains a 16-level stack for last-in, first-out storage. The Data Access Register (9407 DAR) includes three general purpose registers and an adder network which can be used for RAM-resident stack-type functions. A Register Stack (9410 R-stack) provides 16 words of high-speed read/write memory for scratchpad area.

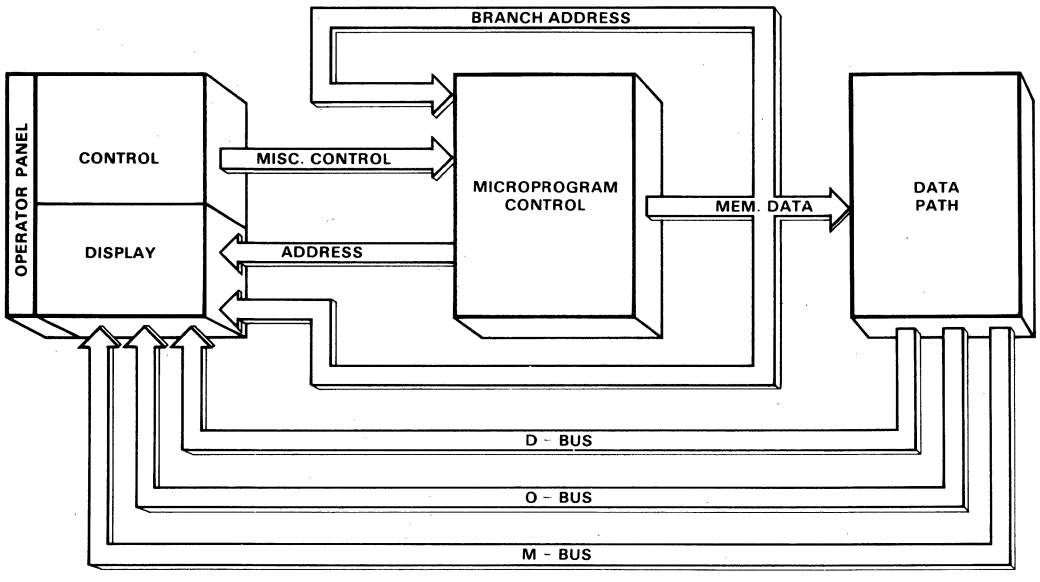


Fig. 3-1 System Block Diagram

The data path illustrates the highly bus-organized configuration possible with Macrologic systems. The instruction inputs of the P-stack, DARs and R-stacks are bussed together to allow instruction field sharing. That is, the same 4-bit field (C4-C7) is used to control the devices, but only one of these devices is allowed to execute an instruction in a microcycle. The K-bus inputs of the DPSs are tied to the eight data input switches on the operator panel so external information can be introduced into the data path. Three other busses are also available — the D-bus, O-bus and M-bus. The D-bus is created by connecting the D-inputs of the DPSs to the corresponding O-outputs of the ALRSs, P-stacks, DARs and R-stacks. The O-bus is the result of connecting the O-outputs of the DPSs to the corresponding D-inputs of the ALRSs, P-stacks, DARs and R-stacks. The M-bus is created by wiring the X outputs of the P-stacks to the X outputs of the DARs.

Except for the DPS, all Macrologic devices in the data path are clocked by a common system clock. Each of these devices has an Execute input ($\overline{EX}$) and the device will not respond to the clock unless ($\overline{EX}$) is LOW. In addition, the 3-state outputs of the devices remain in the high impedance state except when the Output Enable ($\overline{EO}$) is LOW. Thus, during a single clock cycle, information can be routed from any device in the data path to any other device (including itself) by controlling the proper $\overline{EX}$ and $\overline{EO}$ inputs.

The manner in which the individual $\overline{EX}$ and $\overline{EO}$ inputs of the Macrologic devices are controlled is shown in *Figure 3-3*. Bits 0 and 1 of the microprogram control word (C0 and C1) are address inputs to an LS139 1-of-4 decoder. Its active LOW outputs are tied to the Output Enable ($\overline{EO}$) inputs of the ALRSs, DARs, P-stacks and R-stacks, respectively. Therefore, a device can be selected to source data on to the Data Out bus (D-bus) in a clock cycle by appropriately selecting bit 0 and bit 1 of the microprogram control word. Similarly, bit 2 and 3 (C2, C3) of the microprogram control word are inputs to the other half of the LS139 package. Three of the four active LOW outputs are tied to the Execute ($\overline{EX}$) inputs of the DARs, P-stacks, and R-stacks, respectively. Thus, bit 2 and 3 can select one of these three devices to be active and respond to input clocking, or none of them to be activated during a clock cycle. The ALRSs are operated in the general register mode. A two-to-one multiplexer routes the A field inputs of the 9405s to the source and destination registers during the HIGH and LOW periods of the clock respectively. The I_0 input of the most significant ALRS slice is used both to pass instructions and carry information from the least significant ALRS slice.

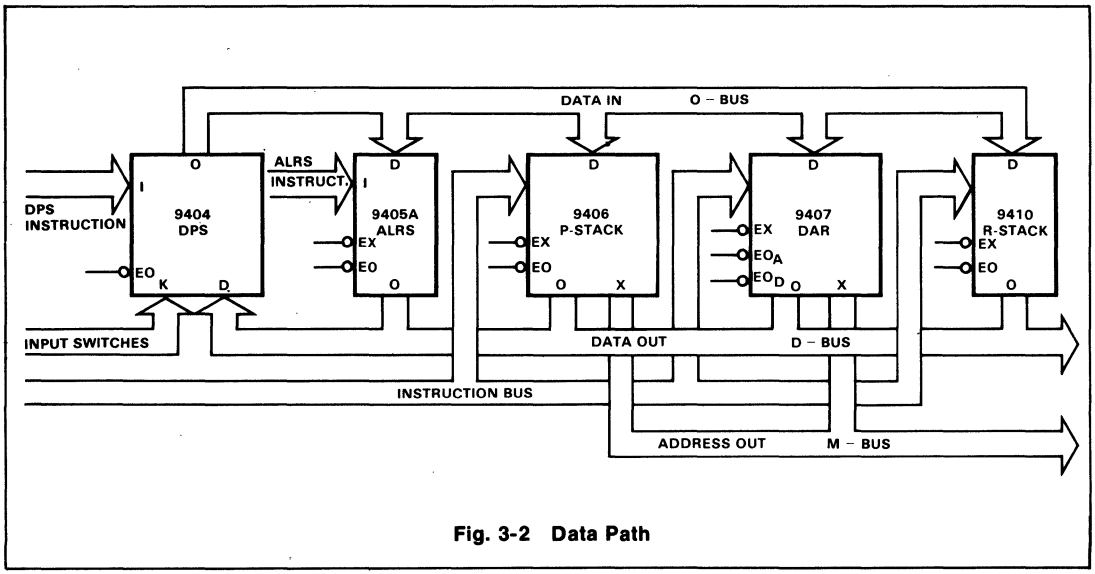


Fig. 3-2 Data Path

To extend the shifting capability in the data path, a two-to-one multiplexer selects data, originating from either a bit in the control memory (C30) or the right and left serial DPS outputs, and enters it into the left and right serial inputs of the DPSs *(Figure 3-4)*.

The select input of the two-to-one multiplexer is controlled by a rotate select bit (C14). If this bit is HIGH during a right shift, the right serial output is shifted into the most significant DPS position which, in effect, makes the instruction a right rotation. For left rotation, the left serial output is shifted into the least significant DPS position. If the rotate select bit (C14) is LOW, then the value of the left or right shift carry in is under microprogram control and determined by control word bit 30 (C30). If C30 is HIGH during a right shift, a "0" will be shifted into the most significant DPS position since the DPS has active LOW inputs.

MICROPROGRAM CONTROL

The microprogram memory is constructed using eight 1K 93L422 fully decoded RAMs organized as 256 words by 32 bits. A 9408 Microprogram Sequencer provides address inputs to the control memory *(Figure 3-5)*.

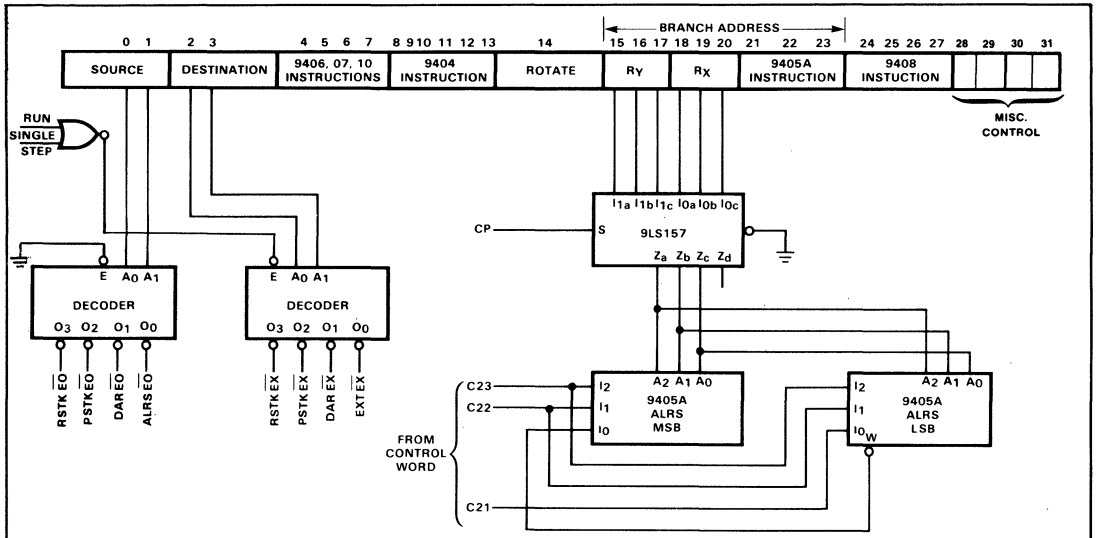


Fig. 3-3 Data In/Out Select and Address Multiplexing in ALRS

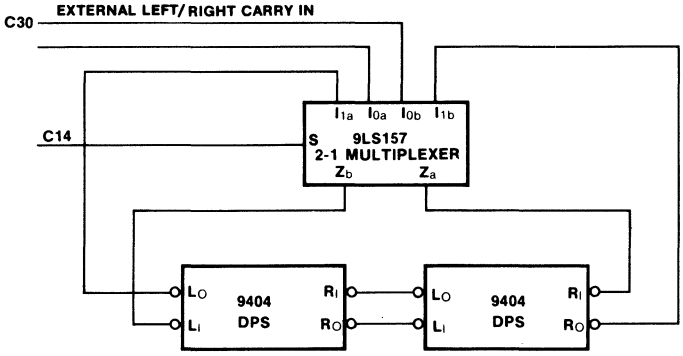


Fig. 3-4 Rotate Control

The clock input of the 9408 is a signal called SEQCK, which is the same as the system clock (CP) that goes to all other Macrologic devices in the data path when the system is in the Run or Single-Step mode. When the system is in the Halt mode, SEQCK is held HIGH and only goes LOW momentarily when either the Address Load switch is depressed, or the Deposit Next or Examine Next switch is depressed and Field Select (FS_1 and FS_0) is "11" (to be described later).

The 9408 Branch Address inputs come from the outputs of two LS157 two-to-one multiplexers whose select input is tied to the VIA_0 output of the 9408. A Branch VIA_0 instruction will select control memory bits 16 to 23 (C16-C23) as the branch address, whereas the Branch VIA_1 instruction will load the microprogram address from the control panel input switches. Actually, the 9408 has the capability of selecting its branch address from up to four different sources through the BRV_0 to BRV_3 instructions. But in the Demonstrator design, only two branch address sources are implemented.

The Multiway switches (M0 to M2) on the control panel feed the 9408 MW_0-MW_2 inputs for multiway branching. The four ALRSs status flags (zero, overflow, carry and negative) are strobed in the 9408 internal test register at the end of each microcycle in which the ALRSs are exercised by an active LOW signal called ALUEXL which also goes to the Execute ($\overline{EX}$) input of the ALRSs. The PLS (Pipeline Select) input is grounded for non-pipeline operation.

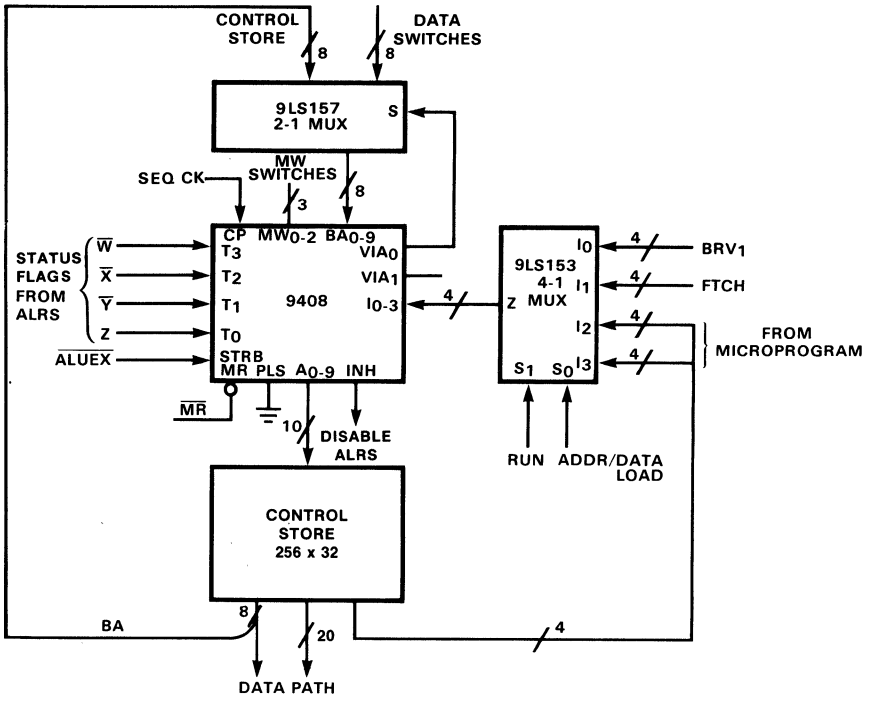


Fig. 3-5 Microprogram Memory and Sequencing Logic

The instruction inputs of the 9408 come from the outputs of two LS153 four-to-one multiplexers. *Table 3-1* depicts the manner in which the various combinations of the instruction inputs are selected. In the Run or Single-Step mode, S_1 of the multiplexer select input is HIGH and the 9408 instruction inputs are entered via the microprogram control memory output bits 24 to 27 (C24-C27). In the Halt mode, S_1 of the multiplexer select input is LOW and S_0 of the multiplexer input causes the 9408 to either increment the microprogram address by one (FETCH) or load the microprogram address from the control panel input switches (BRV_1) when the 9408 is clocked. Normally, the clock input of the 9408 is disabled (held HIGH) and the S_0 input of the LS153 is HIGH, forcing the FETCH instruction of the 9408 instruction inputs. When Field Select (FS_1 and FS_0) is "11" and either the Examine Next or Deposit Next switch is activated, the 9408 is clocked once. This increments the microprogram address by one, allowing the next control word to be examined or loaded. Operation of the Load Address switch momentarily brings the S_0 input of the LS153 LOW, forcing the BRV_1 instruction on the 9408 instruction inputs, and selects the input switches as 9408 Branch Address inputs. The 9408 is again clocked once to match its address outputs to the input switch data.

OPERATOR PANEL

The operator panel consists of two sections — control and display. The control section provides the operator interface to the control memory and data path through a number of toggle switches. The display allows the operator to monitor the microprogram address and data, ALU status flags and the various system busses through LEDs. In addition, a number of turret terminals are provided to allow easy access to some important system signals.

Display

The display consists of two rows of eight LEDs which read the microprogram address and data; a row of eight bus status LEDs, four ALU status LEDs, two Field Select LEDs and a RUN indicator LED. Since the microprogram control word is 32 bits wide, two control inputs (FS_1 and FS_0) select four separate fields to be displayed by a single set of eight data LEDs. Setting FS_1 and FS_0 to "00" produces the left most (most significant) eight bits of a control word. A setting of "01" produces the next eight bits, "10" the next, and "11" the right most (least significant) 8-bit field. The ALRS flags (zero, negative, carry and overflow) are latched into the four status LEDs so that ALU status can be continuously monitored.

Control

The panel control section, contains a group of eight toggle switches (SW0-SW7) for Branch Address or Data input to the system, as well as a number of miscellaneous control switches. The Clock Select switch selects between an internal 2 MHz oscillator or an external clock provided by

S_1	S_0	I_3	I_2	I_1	I_0	MEANING
0	0	0	1	0	1	BRV_1
0	1	0	0	1	0	FETCH
1	0	C24	C25	C26	C27	Controlled by
1	1	C24	C25	C26	C27	Microprogram

$S_1 = 0$ Halt
$S_1 = 1$ Run or Single Step
$S_0 = 0$ Address Load
$S_0 = 1$ Data Load

Table 3-1 9408 Instruction Inputs Control

the user. If an external clock (such as a pulse generator) is used, it should be attached to the EXT-CLK IN turret terminal on the board. A group of three Multiway switches (M0-M2) provides the multiway inputs to the 9408 for eight-way branching. Two Data Bus Select switches (DS1, DS0) select either a lamp test or the contents of one of three system busses (O, D, or M-bus) for display by the eight data-bus LEDs as illustrated by *Table 3-2*. The Master Reset switch halts the system and clears the microprogram address and field to zero.

The Run/Halt switch is a 2-function toggle switch that selects either the Run or Halt mode. If the processor is halted, pressing this switch returns the processor to the Run mode. The processor then executes the microprogram beginning with the instruction contained in the memory location pointed to by the microprogram address. It continues in this fashion until the Run/Halt switch is pressed again; however, the processor halts only after completion of the current microinstruction.

While the processor is in the Halt mode, a number of other switch functions can be activated. The Load Address switch loads input switch data as the microprogram address. The Deposit switch writes the 8-bit words, at the input switches, into the field of the memory location addressed by the 9408 address and Field Select (FS_1 and FS_0) outputs. The Deposit Next switch operates similarly to Deposit; however, the field is incremented by one before the input switch data is written into memory. Moreover, if Field Select is "11," the field is recirculated to "00" and the 9408 address outputs are incremented by one before the input switch data is written into memory. Thus, the user can load a microprogram eight bits at a time without laboriously loading addresses by repeatedly setting up data switches and then pressing the Deposit Next switch.

The Examine Next switch displays the data in the next field by lighting the appropriate LEDs, and allows the operator to examine memory data field by field and location by location. The microprogram address LEDs display the memory location of the next processor instruction. Pressing the Single Step switch executes that instruction and then halts the processor. The Single Step switch is especially useful in program debug.

A number of turret terminals provide convenient access to some system signals as test points for viewing on the oscilloscope. The CP test point connects directly to the system clock line of the Demonstrator and represents the actual clock signal applied to the Macrologic devices in the data path. The ALUEXL test point is the LOW active signal which goes to the Execute ($\overline{EX}$) input of the ALRS. The ADDR SYNC test point outputs a HIGH pulse whenever the microprogram address matches the input switch word. In this manner, the input switches can be used to select the microprogram address at which a sync pulse is desired. This provides a handy signal for use as a reference trace on the oscilloscope and signals in the vicinity of the synchronized instruction can be examined by placing the microprogram in a loop. The C31 test point is tied directly to control word bit 31, which is a spare bit at this time. However, it can be used as an additional marker to identify a microinstruction or provide another reference signal on the scope. A Ground point is also brought out to a turret terminal next to the ADDR SYNC test point so that the ground clip of an oscilloscope probe can be attached to it.

DATA BUS SELECT		BUS SELECTION
DS1	DS0	
0	0	O-Bus
0	1	D-Bus
1	0	M-Bus
1	1	Lamp Test

Table 3-2 Data Bus Select Switches Operation

CONTROL WORD FORMAT

Choosing the width and format of the microprogram control word is a task which involves tradeoff between system performance and memory cost. A format which dedicates separate fields for each individual control function allows many independent operations to be done in one microcycle but also yields the widest control word. Usually, some sort of field sharing is implemented to reduce the control word width without sacrificing too much performance. The Macrologic Demonstrator uses a 32-bit control word reduced by the following overlapping and encoding techniques.

The control inputs for the ALRS and the Branch Address inputs of the 9408 are overlapped since most program structures consist of more fetches than branches. The address for the 9408 Microprogram Sequencer is implicit during Fetch (FTCH) and Return from Subroutine (RTS), which suggests that the branch address field of the 9408 can be overlapped with the instruction input field of another Macrologic element in the system. In this case, the 8-bit branch address field can be conveniently shared with the instruction and register fields of the ALRS. The 9408 Inhibit (INH) output simplifies field overlap; the $\overline{EX}$ input of the ALRS is simply disabled except during a FTCH or RTS.

Since the O-bus is 3-state, only one of the four Macrologic elements can drive this bus at any time. Thus the four O-bus sources can be encoded by two bits. Since the likelihood is small that more than one Macrologic element in the system will need to operate on the same input during the same microcycle, the four D-bus destinations can also be encoded by two bits. Actually, there are only three Macrologic device destinations and four possible permutations of two bits, which leaves a 'free' state "00." This state can be used to select an external device or to deactivate all three Macrologic elements. Note that the ALRS execution is controlled by the INH output of the 9408 and is unaffected by this field. The result is a 32-bit control word with field definition as shown in *Table 3-3.*

BITS	FUNCTION
C0 — C1	O-bus source. These bits select the device driving the input bus of the DPS by activating the proper Output Enable ($\overline{EO}$) input. "00" — ALRS, "01" — DAR, "10" P-stack, "11" — R-stack
C2 — C3	D-bus destination. These bits specify device to be activated and respond to C4 — C7 by enabling the proper Execute ($\overline{EX}$) input. "00" — EXT*, "01" — DAR, "10" — P-stack, "11" — R-stack
C4 — C7	Shared between P-stack, DAR and R-stack, these bits are Instruction inputs to the P-stack and DAR and Address inputs to the R-stack.
C8 — C13	Instruction inputs to the DPS.
C14	Rotate control. Specifies source of carry in (external or rotate) during DPS shifts.
C15 — C23	Shared between the 9408 Sequencer and ALRS. When the 9408 instruction (C24 — C27) is neither a Fetch or a Return, these bits are the Branch Address. Otherwise, C15 — C17 select one ALRS register for destination of ALU operations, C18 — C20 select one ALRS register as source of ALU operations, C21 — C23 become ALRS Instruction inputs.
C24 — C27	9408 Sequencer Instruction inputs.
C28	9407 X-bus enable input. When the 9406 is executing a FETCH instruction, C28 should be held HIGH for microcycle to prevent both the 9406 and 9407 from driving the M-bus at the same time.
C29	9406 and 9407 carry in input.
C30	9404 carry in input for shift function. If rotate control (C14) is LOW, the value of C30 (active LOW) will be shifted into the least or most significant position of the DPS following a left or right shift, respectively.
C31	Spare. This bit is brought out to a turret terminal and can be used as a marker to identify one or a group of microinstructions.

*EXT = External, non-selected

Table 3-3 Control Word Format

MISCELLANEOUS

The Demonstrator is pretty much self-contained. The only additional equipment needed for its operation is a 5 V power supply capable of delivering about 2 A of current to be attached to the V_{CC} and GND turret terminals in the lower left hand corner of the board.

Instead of hand loading the microprogram through input switches, a user who has an S-100 bus 8080 based system can choose to have the control store of the Demonstrator loaded by the 8080. The Demonstrator card edge connector plugs directly into the S-100 bus. A μA7805 voltage regulator in TO-220 package is used to obtain V_{CC} from the 8 V which come from pin 1 of the S-100 bus. A 5 Ω, 2 W resistor is required between the input and output terminal of the regulator to provide enough current for the board.

OPERATING PROCEDURE FOR CONTROL PANEL OF MACROLOGIC DEMONSTRATOR

The following operations are possible when the system is in the Halt mode. To place the system in the Halt mode, use either the Run/Halt switch or press the Master Reset switch momentarily.

Loading In Microprogram

1. Press Master Reset. Make sure system is in Halt mode by verifying Run light is off.
2. Set up starting address via the eight data input switches.
3. Hit Load Address switch momentarily and verify the address lights show the same data as the input switch settings.
4. Verify Field Select (FS_1 and FS_0) lights = "00." Set up data for first field in data input switches.
5. Depress Deposit switch and verify the data lights show the same data as the input switch setting.
6. Set up data for the next field through the input switches.
7. Depress Deposit Next switch and verify:
   a. The Field Select (FS_1, FS_0) lights are incremented by one.
   b. If the Field Select lights = "11," the address light will be incremented by one and the Field Select lights will go to "00."
   c. The data lights are the same as the input switch settings.
8. Repeat steps 6 and 7 to input data sequentially, field by field, location by location.

Verifying Program

1. Press Master Reset to get into the Halt mode.
2. Set up desired starting address of microprogram by input switches. Press Load Address switch momentarily and verify the address lights show the same data as the input switch settings.
3. By continuing depressing the Examine Next switch, one can sequentially examine the contents of the microprogram, 8-bit field at a time, location by location.

Modifying Program

1. Hit Master Reset to clear out address and field lights.
2. Enter starting address via input switches and depress Load Address switch.
3. Get to desired field using Examine Next switch.
4. Verify old data is displayed on data lights. Set up new desired data through input data switches.
5. Hit Deposit switch and verify new data is deposited.
6. Repeat above steps for other fields to be modified or use Examine Next switch to advance to the next field to be modified and repeat step 4 and 5.

Single Stepping Microprogram

1. Make sure system is in Halt mode. Set up starting microprogram address via the eight data input switches.
2. Press Load Address switch and verify the address lights display the same data as input switches.
3. Depressing Single Step switch momentarily will cause the microinstruction pointed to by the microprogram address LEDs to be executed. The system will then be halted.
4. Repeat step 3 to execute the microprogram one microinstruction at a time.

Running the Microprogram

1. Set up starting address via input switches and hit Load Address switch.
2. Depress Halt/Run switch and the RUN light will come on, verifying the system has begun execution starting from the microinstruction specified in the microprogram address.

Loading the Microprogram Memory Through the S-100 Bus

The Macrologic Demonstrator has the provision for an S-100 bus system to load its control memory. The write signal from the 8080 Processor (PWRL) is simply made to act like the Deposit Next switch. The Demonstrator board decodes the two higher order bits of the 8080 address bus and makes sure they are equal to "11" before its control memory is loaded, i.e., the Demonstrator can have a hexadecimal memory address of CXXX up to FXXX. To load the microprogram memory, 8-bit field at a time, do the following:

1. Halt the 8080 system. Plug the Macrologic Demonstrator into the S-100 bus connector.
2. Press Master Reset and verify the address and field lights are zero and the system is halted.
3. Set up the starting microprogram address via the input switches and press the Load Address switch.
4. Set up the first field of the microprogram through the input switches. Depress Deposit momentarily. Verify the data lights are the same as the input switch settings.
5. Put all eight data switches in the Up position.
6. Any subsequent write by the 8080 Processor to memory address CXXX through FXXX will increment the field by one first before writing the data, i.e., it generates the same action as if the Deposit Next switch is depressed.

This gives the user an alternative to loading the microprogram memory through input switches. A user can now, say, read in a microprogram through the paper tape reader of the 8080 system and transfer it to the control memory of the Macrologic Demonstrator.

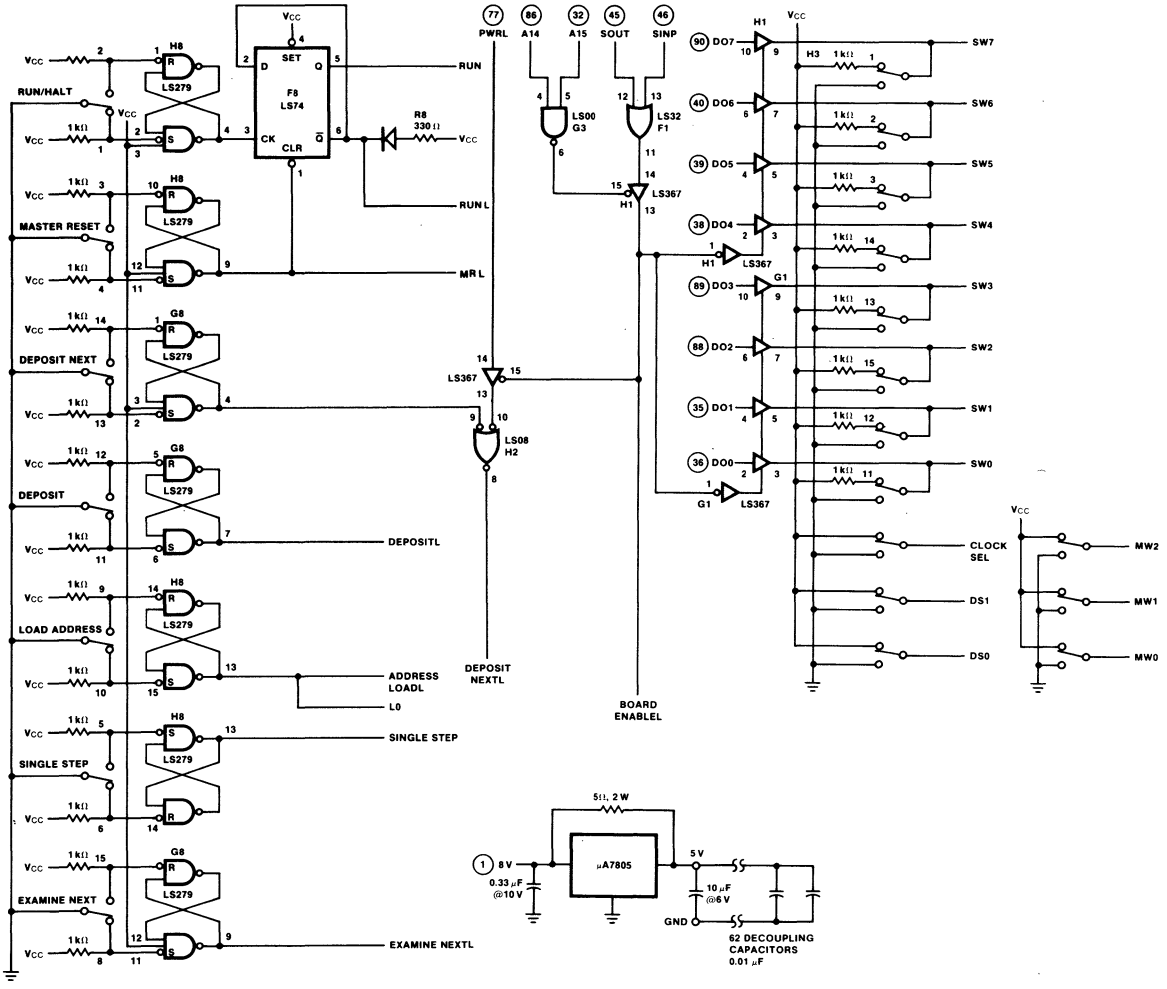


Fig. 3-6 Macrologic Control Panel Switches Schematic

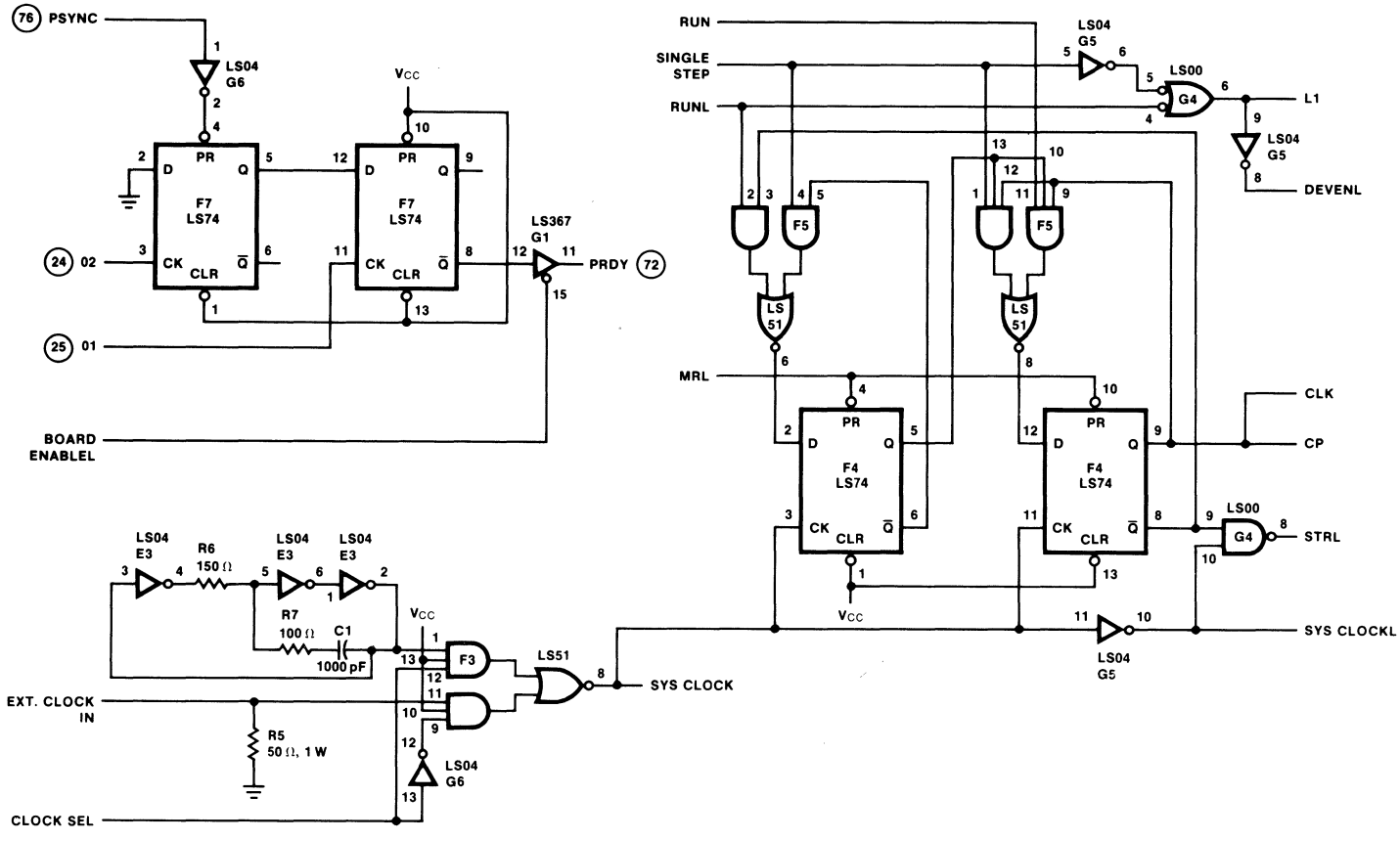


Fig. 3-7 Clock & Single Step Logic Schematic

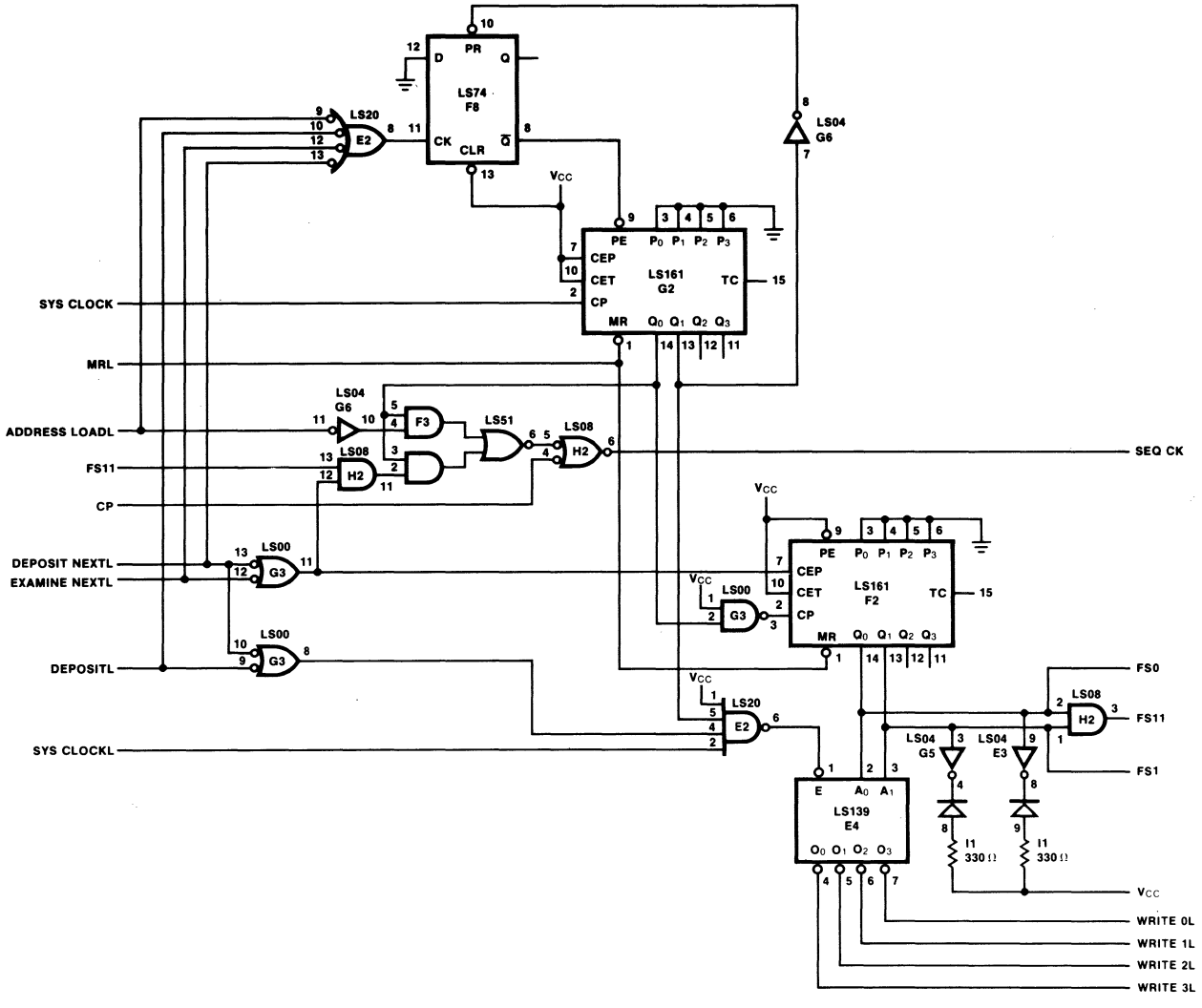


Fig. 3-8 Examine, Deposit, and Address Load Logic Schematic

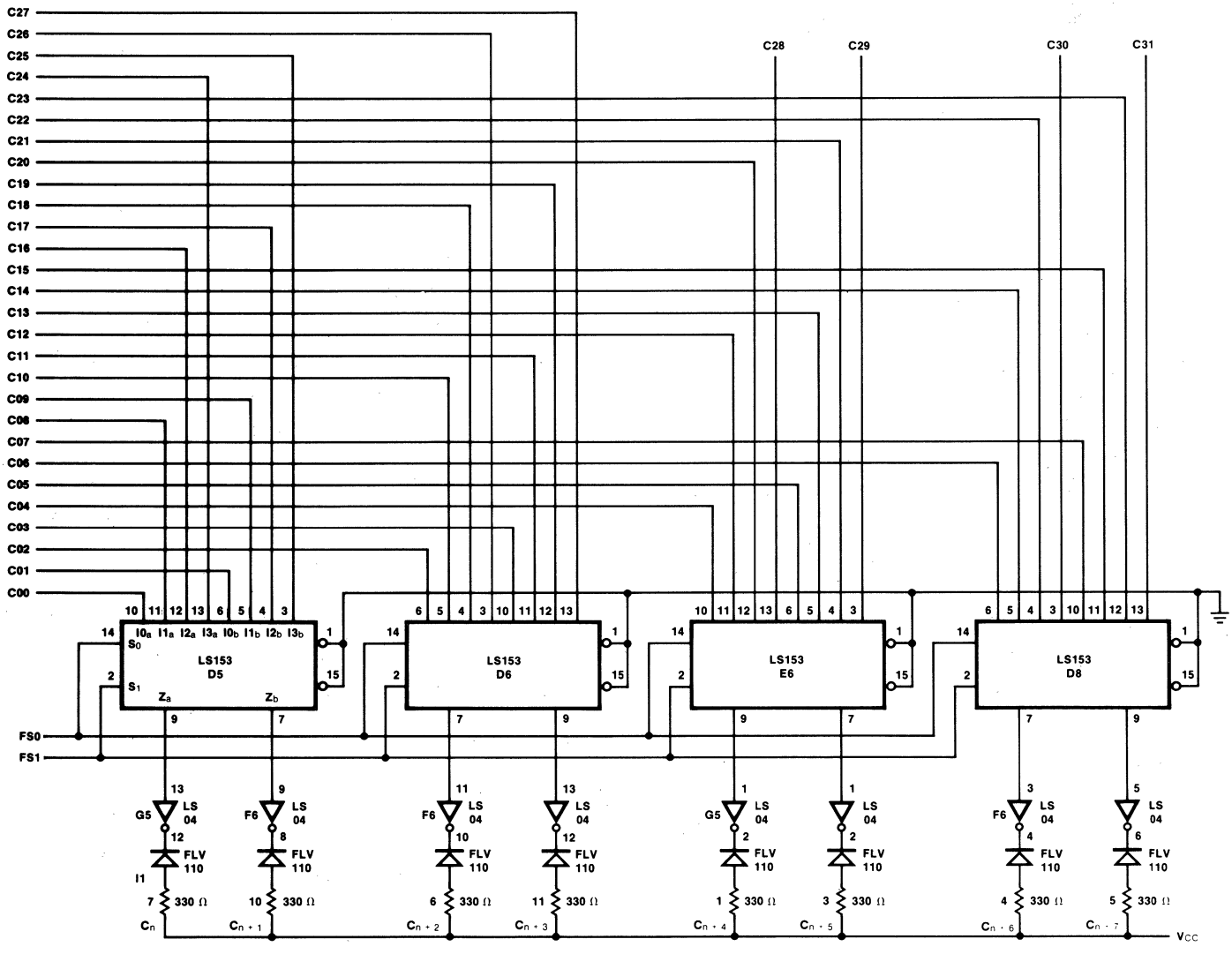


Fig. 3-9 Microprocessor Data Display Schematic

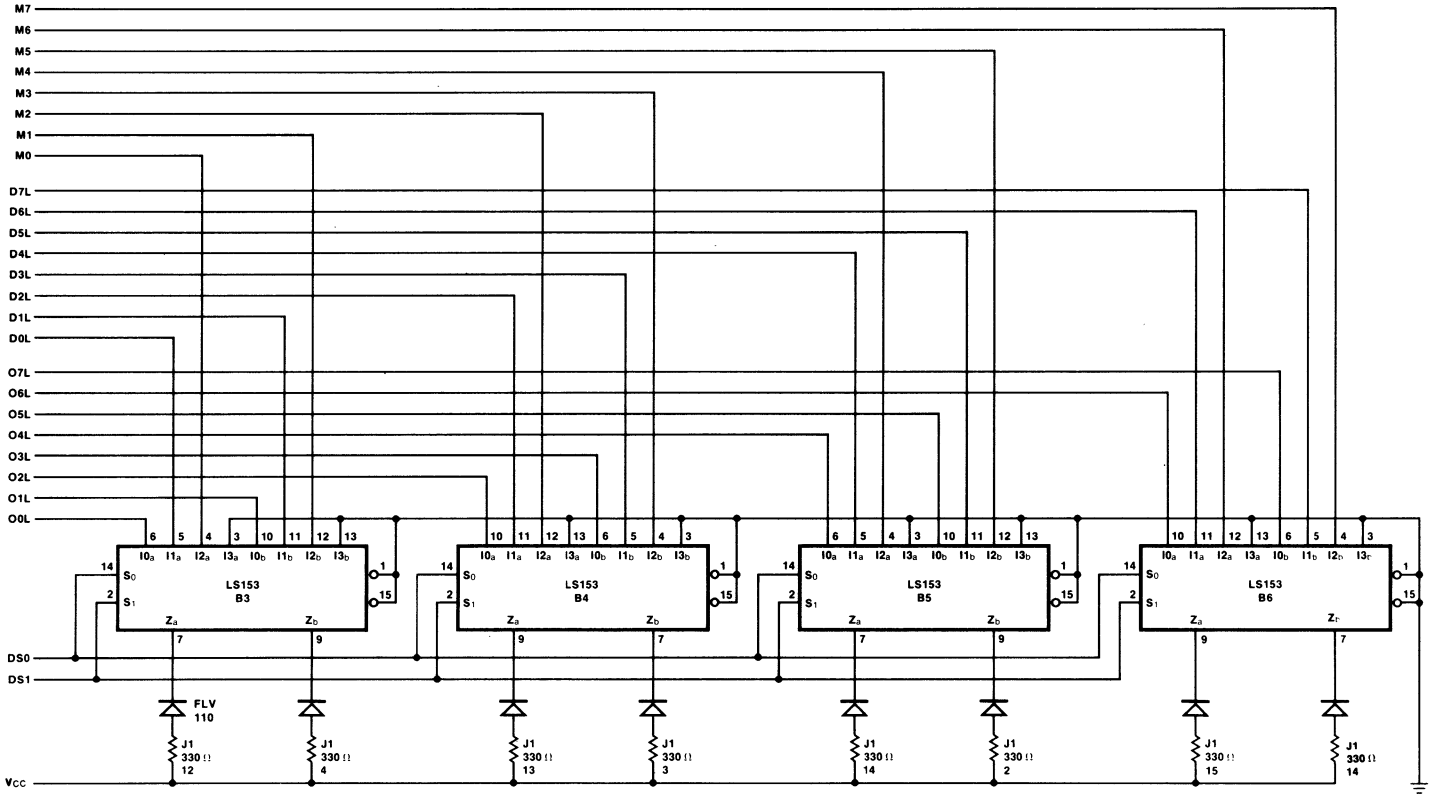


Fig. 3-10 Control Panel Data Bus Select and Display Schematic

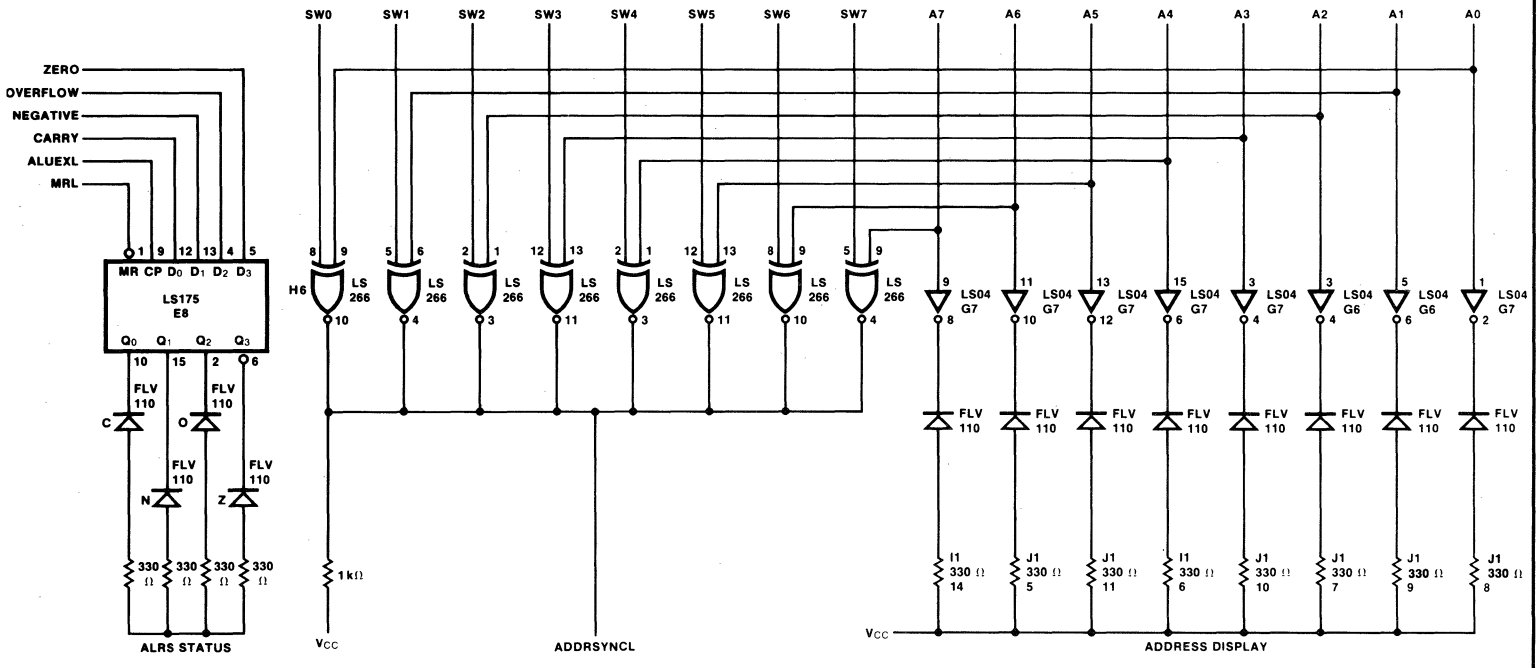


Fig. 3-11 Address Sync, ALRs Flags and Address Display Schematic

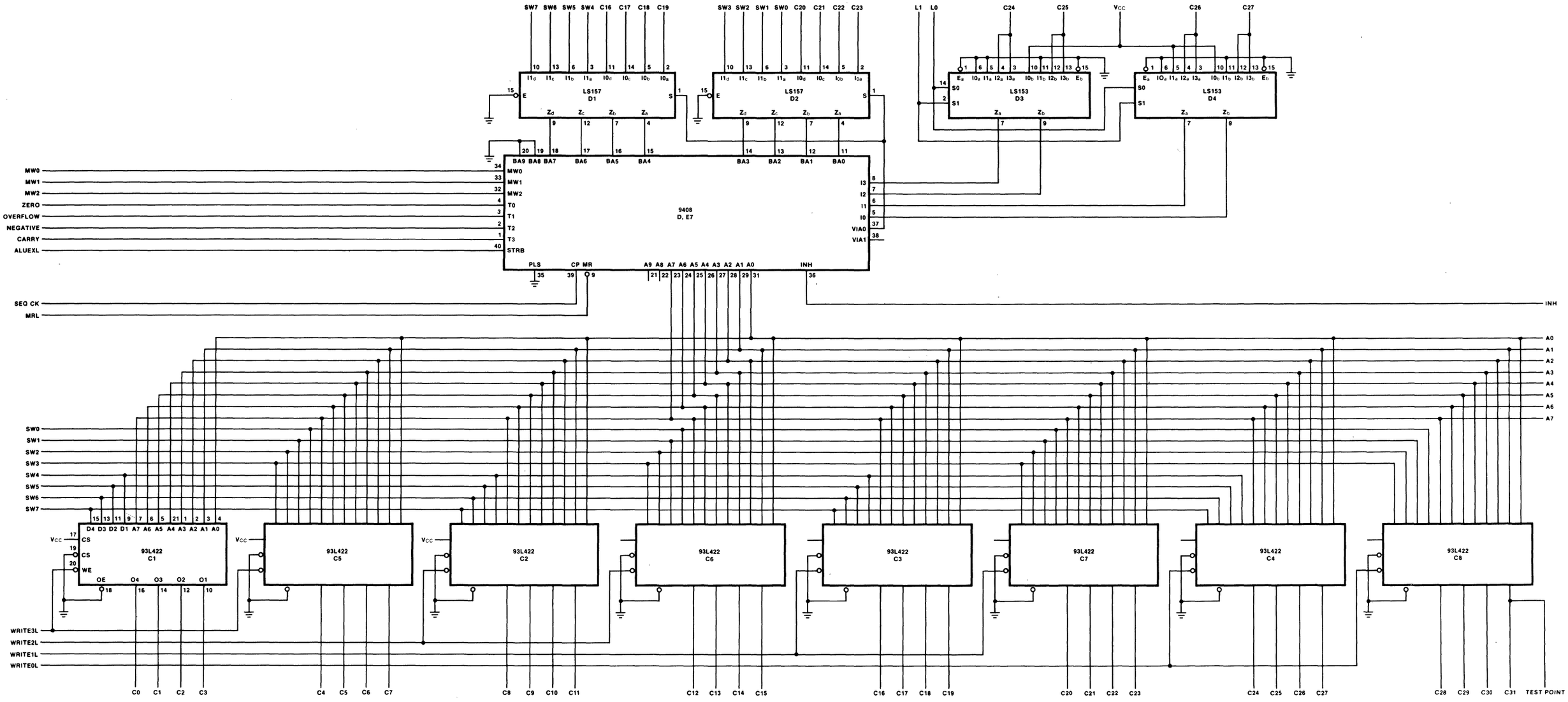


Fig. 3-12 Micro Control Store and Sequencing Logic Schematic

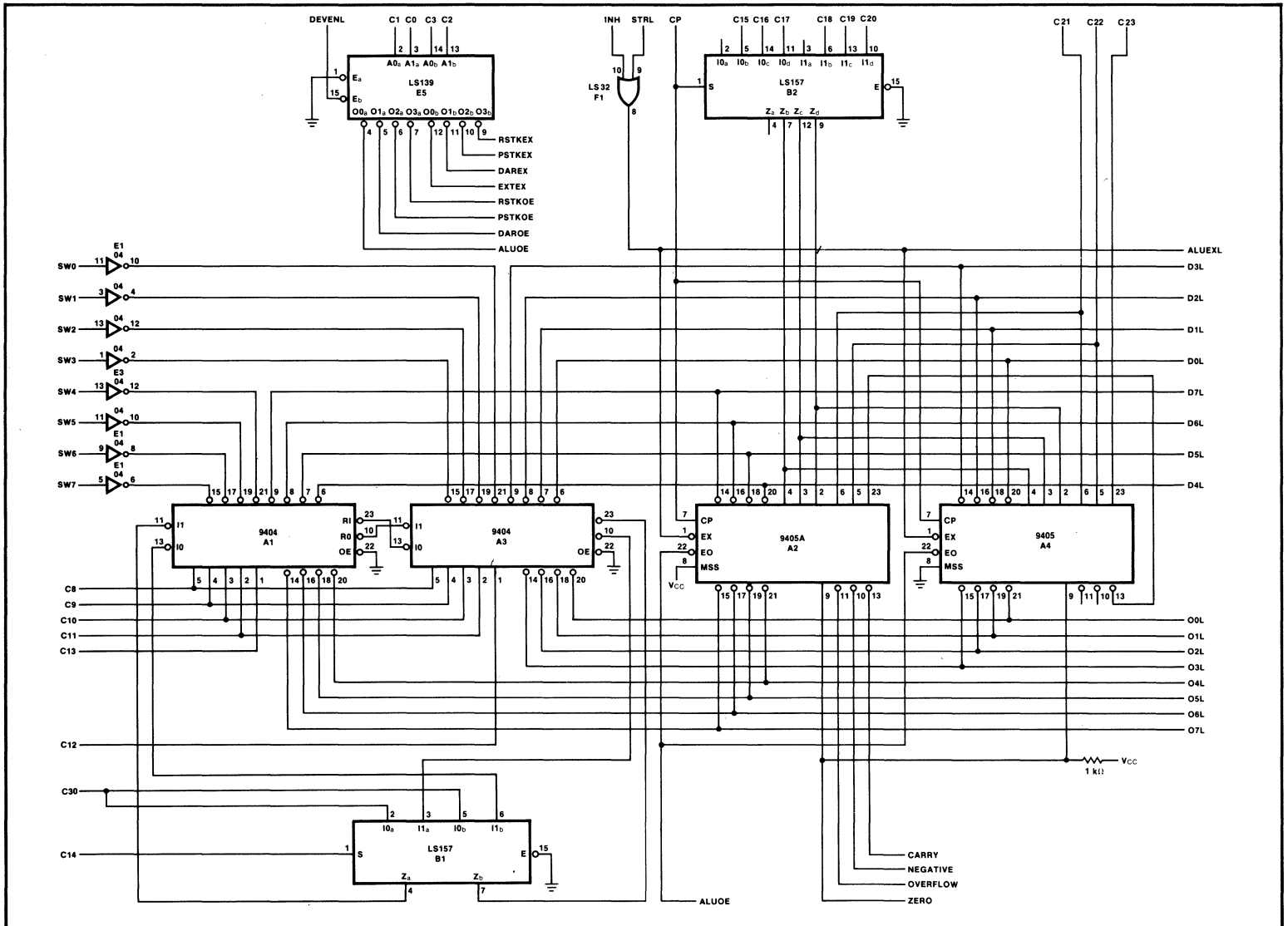


Fig. 3-13 Data Path I Schematic

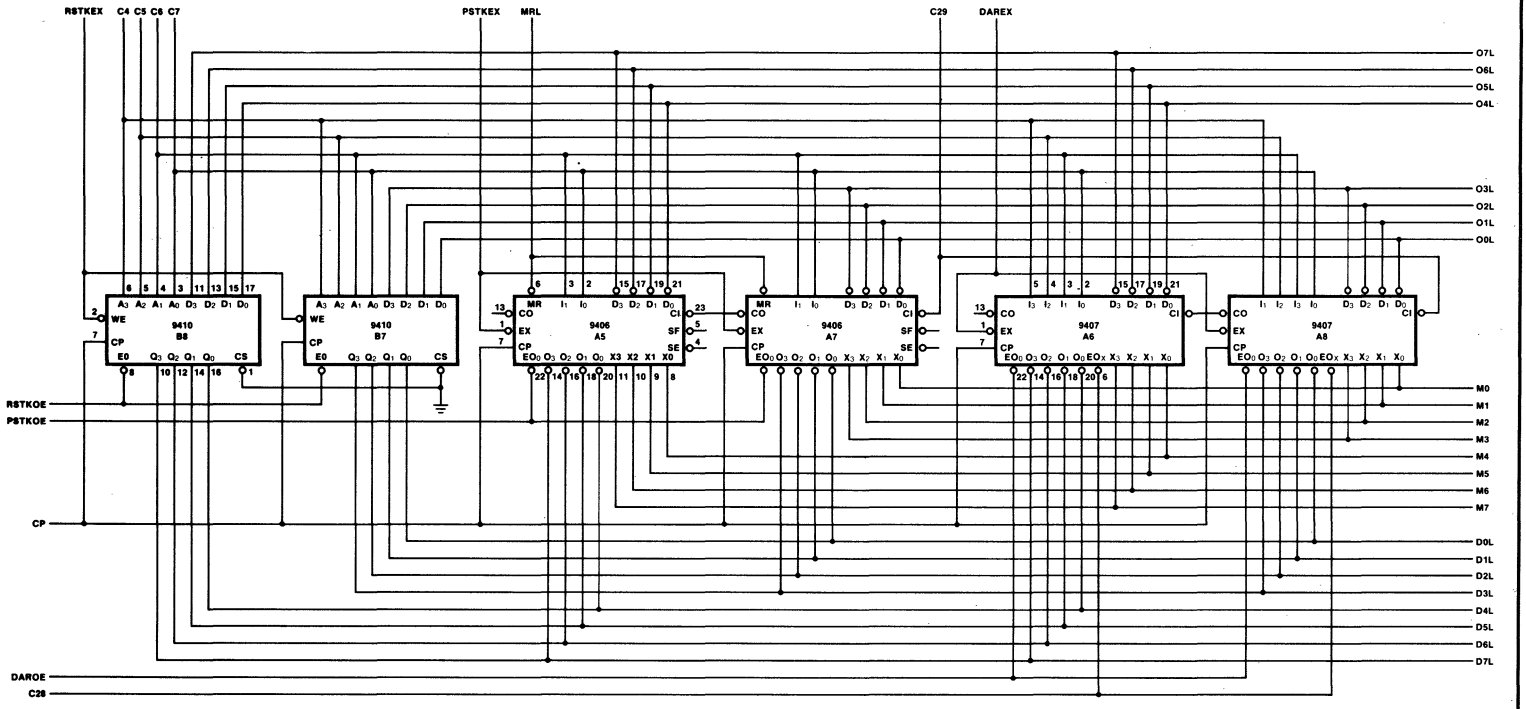


Fig. 3-14 Data Path II Schematic

Section 4
PROGRAMMING EXAMPLES AND EXERCISES

The previous section described the data path architecture and the 32-bit microinstruction format of the Macrologic Demonstrator. This section will show some programming examples and exercises geared to demonstrate the uses of the Macrologic devices and to involve the user in the microprogramming of these devices. The binary information which specifies the control store address and contents can be realized by either hand-coding the various fields of each control word or using a microprogram assembler. (Fairchild offers a choice of assembler software, the microprogram assembler and DAPL, available through time sharing networks.)

After a microprogram is loaded, the user can single step through it. In this mode, the user can observe the various states of the machine at each microcycle. Alternatively, the user can evaluate the dynamic behavior of the machine by putting it in the Run mode. This allows the Macrologic devices to be tested under operating conditions and some switching characteristics to be measured by an oscilloscope.

Table 4-1 provides a summary of the instruction set definitions for the various Macrologic devices. However, users should refer to the Macrologic Data Book for detailed device information.

9404

INPUTS					OUTPUTS				FUNCTION	INPUTS					OUTPUTS						FUNCTION
I_4	I_3	I_2	I_1	I_0	$\overline{O_3}$	$\overline{O_2}$	$\overline{O_1}$	$\overline{O_0}$		I_4	I_3	I_2	I_1	I_0	$\overline{LO}$	$\overline{O_3}$	$\overline{O_2}$	$\overline{O_1}$	$\overline{O_0}$	$\overline{RO}$	
L	L	L	L	L	L	L	L	L	Byte Mask	H	L	L	L	L	$\overline{RI}$	$\overline{RI}$	$\overline{RI}$	$\overline{RI}$	$\overline{RI}$		K-Bus Sign Extend
L	L	L	L	H	H	H	H	H	Byte Mask	H	L	L	L	H	$\overline{K_3}$	$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$		K-Bus Sign Extend
L	L	L	H	L	L	L	L	H	Minus "2" in 2s Comp	H	L	L	H	L	$\overline{RI}$	$\overline{RI}$	$\overline{RI}$	$\overline{RI}$	$\overline{RI}$		D-Bus Sign Extend
L	L	L	H	H	L	L	L	L	Minus "1" in 2s Comp	H	L	L	H	H	$\overline{D_3}$	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$		D-Bus Sign Extend
L	L	H	L	L	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$	Byte Mask, D-Bus	H	L	H	L	L	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$	$\overline{RI}$		D-Bus Shift Left
L	L	H	L	H	H	H	H	H	Byte Mask, D-Bus	H	L	H	L	H	$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$	$\overline{RI}$		K-Bus Shift Left
L	L	H	H	L	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$	Byte Mask, D-Bus	H	L	H	H	L		$\overline{LI}$	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$	D-Bus Shift Right
L	L	H	H	H	L	L	L	L	Byte Mask, D-Bus	H	L	H	H	H		$\overline{D_3}$	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$	D-Bus Shift Right Arith
L	H	L	L	L	L	H	H	H	Negative Byte Sign Mask	H	H	L	L	L		$\overline{LI}$	$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$	K-Bus Shift Right
L	H	L	L	H	H	H	H	H	Positive Byte Sign Mask	H	H	L	L	H		$\overline{K_3}$	$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$	K-Bus Shift Right Arith
L	H	L	H	L	$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$	Byte Mask, K-Bus	H	H	L	H	L		$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$		Byte Mask, K-Bus
L	H	L	H	H	L	L	L	L	Byte Mask, K-Bus	H	H	L	H	H		H	H	H	H		Byte Mask, K-Bus
L	H	H	L	L	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$	Load Byte	H	H	H	L	L		D_3	D_2	D_1	D_0		Complement D-Bus
L	H	H	L	H	$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$	Load Byte	H	H	H	L	H		K_3	K_2	K_1	K_0		Complement K-Bus
L	H	H	H	L	H	H	H	L	Plus "1"	H	H	H	H	L							Undefined (Reserved)
L	H	H	H	H	H	H	H	H	Zero	H	H	H	H	H							Undefined (Reserved)

Comp = Complement
Arith = Arithmetic

Table 4-1 Instruction Sets for Macrologic Devices

9405A

INPUTS I_2	I_1	I_0	ACCUMULATOR MODE ($R_x = R_y$)		GENERAL REGISTER MODE ($R_x \neq R_y$)	
L	L	L	R_x *plus* $\overline{\text{D-Bus}}$ *plus* 1 → R_x	Accumulate and Increment	R_x *plus* $\overline{\text{D-Bus}}$ *plus* 1 → R_y	Add with Carry
L	L	H	R_x *plus* $\overline{\text{D-Bus}}$ → R_x	Accumulate	R_x *plus* $\overline{\text{D-Bus}}$ → R_y	Add
L	H	L	R_x $\overline{\text{D-Bus}}$ → R_x	Logical AND	R_x $\overline{\text{D-Bus}}$ → R_y	Logical AND
L	H	H	$\overline{\text{D-Bus}}$ → R_x	Load	$\overline{\text{D-Bus}}$ → R_y	Load
H	L	L	R_x → Output Register	Read	R_x → R_y	Transfer
H	L	H	R_x + $\overline{\text{D-Bus}}$ → R_x	Logical OR	R_x + $\overline{\text{D-Bus}}$ → R_y	Logical OR
H	H	L	$R_x \oplus \overline{\text{D-Bus}}$ → R_x	Exclusive OR	$R_x \oplus \overline{\text{D-Bus}}$ → R_y	Exclusive OR
H	H	H	D-Bus → R_x	Load Complement	D-Bus → R_y	Load Complement

NOTES:
R_x is the RAM location addressed by $A_0 - A_2$ when CP is HIGH.
R_y is the RAM location addressed by $A_0 - A_2$ when CP is LOW.
The result of any operation is always loaded into the Output Register at the end of the cycle provided that $\overline{EX}$ is LOW.

9406

INPUTS I_1	I_0	INSTRUCTION	INTERNAL OPERATION	X-BUS	O-BUS (WITH EO_0 LOW)
L	L	Return (Pop)	Decrement Stack Pointer	Disabled	Depending on the relative timing of $\overline{EX}$ and CP, the outputs will reflect the current program counter or the new value while CP is LOW. When CP goes HIGH again, the output will reflect the new value.
L	H	Branch (Load PC)	Load D-Bus into Current Program Counter Location	Disabled	Current Program Counter until CP goes HIGH again, then updated with newly entered PC value.
H	L	Call (Push)	Increment Stack Pointer and Load D-Bus into New Program Counter Location	Disabled	Depending on the relative timing of $\overline{EX}$ and CP, the outputs will reflect the current program counter or the previous contents of the incremented SP location. When CP goes HIGH again, the outputs will reflect the newly entered PC value.
H	H	Fetch (Increment PC)	Increment Current Program Counter if $\overline{CI}$ is LOW	Current Program Counter while both CP and $\overline{EX}$ are LOW, disabled while CP or $\overline{EX}$ is HIGH	Current Program Counter until CP goes HIGH again, then updated with incremented PC value.

9407

INPUTS I_3	I_2	I_1	I_0	COMBINATORIAL FUNCTION AVAILABLE ON THE X-BUS	SEQUENTIAL FUNCTION OCCURRING ON THE NEXT RISING CP EDGE
L	L	L	L	R_0	R_0 *plus* $\bar{D}$ *plus* $\overline{CI}$ → R_0 and Output Register
L	L	L	H	R_0 *plus* $\bar{D}$ *plus* $\overline{CI}$	
L	L	H	L	R_0	R_0 *plus* $\bar{D}$ *plus* $\overline{CI}$ → R_1 and Output Register
L	L	H	H	R_0 *plus* $\bar{D}$ *plus* $\overline{CI}$	
L	H	L	L	R_0	R_0 *plus* $\bar{D}$ *plus* $\overline{CI}$ → R_2 and Output Register
L	H	L	H	R_0 *plus* $\bar{D}$ *plus* $\overline{CI}$	
L	H	H	L	R_1	R_1 *plus* $\bar{D}$ *plus* $\overline{CI}$ → R_1 and Output Register
L	H	H	H	R_1 *plus* $\bar{D}$ *plus* $\overline{CI}$	
H	L	L	L	R_2	$\bar{D}$ *plus* $\overline{CI}$ → R_2 and Output Register
H	L	L	H	$\bar{D}$ *plus* $\overline{CI}$	
H	L	H	L	R_0	$\bar{D}$ *plus* $\overline{CI}$ → R_0 and Output Register
H	L	H	H	$\bar{D}$ *plus* $\overline{CI}$	
H	H	L	L	R_2	R_2 *plus* $\bar{D}$ *plus* $\overline{CI}$ → R_2 and Output Register
H	H	L	H	R_2 *plus* $\bar{D}$ *plus* $\overline{CI}$	
H	H	H	L	R_1	$\bar{D}$ *plus* $\overline{CI}$ → R_1 and Output Register
H	H	H	H	$\bar{D}$ *plus* $\overline{CI}$	

Table 4-1 Instruction Sets for Macrologic Devices (Cont.)

	MNEMONIC	DEFINITION	INPUTS $I_3I_2I_1I_0$	$T_3T_2T_1T_0$	$O_9O_8O_7 \cdots O_2O_1O_0$	VIA_1VIA_0	INH	DESCRIPTION OF OPERATION
Unconditional Branch Instructions	BRV_0	Branch VIA_0	L H L L	X X X X	BA_9 BA_8--BA_1 BA_0	L L	H	BA_0 - $BA_9 \rightarrow PC$
	BRV_1	Branch VIA_1	L H L H	X X X X	BA_9 BA_8--BA_1 BA_0	L H	H	BA_0 - $BA_9 \rightarrow PC$
	BRV_2	Branch VIA_2	L H H L	X X X X	BA_9 BA_8--BA_1 BA_0	H L	H	BA_0 - $BA_9 \rightarrow PC$
	BRV_3	Branch VIA_3	L H H H	X X X X	BA_9 BA_8--BA_1 BA_0	H H	H	BA_0 - $BA_9 \rightarrow PC$
	BMW	Branch Multiway	L L H H	X X X X	BA_9 BA_3--MW_2 MW_0	L L	H	MW_0 - MW_2; BA_3 - $BA_9 \rightarrow PC$
	BSR	Branch to Subroutine	L L L H	X X X X	BA_9 BA_8--BA_1 BA_0	L L	H	BA_0 - $BA_9 \rightarrow PC$ & Push the Stack
Conditional Branch Instructions	BTH_0	Branch on T_0 HIGH	H H L L	X X X H X X X L	BA_9 BA_8--BA_1 BA_0 PC+1	L L	H	If Test Register 0 is HIGH: BA_0 - $BA_9 \rightarrow PC$ If Test Register 0 is LOW: PC + 1 $\rightarrow$ PC
	BTH_1	Branch on T_1 HIGH	H H L H	X X H X X X L X	BA_9 BA_8--BA_1 BA_0 PC+1	L L	H	If Test Register 1 is HIGH: BA_0 - $BA_9 \rightarrow PC$ If Test Register 1 is LOW: PC+1 $\rightarrow$ PC
	BTH_2	Branch on T_2 HIGH	H H H L	X H X X X L X X	BA_9 BA_8--BA_1 BA_0 PC+1	L L	H	If Test Register 2 is HIGH: BA_0 - $BA_9 \rightarrow PC$ If Test Register 2 is LOW: PC+1 $\rightarrow$ PC
	BTH_3	Branch on T_3 HIGH	H H H H	H X X X L X X X	BA_9 BA_8--BA_1 BA_0 PC+1	L L	H	If Test Register 3 is HIGH: BA_0 - $BA_9 \rightarrow PC$ If Test Register 3 is LOW: PC+1 $\rightarrow$ PC
	BTL_0	Branch on T_0 LOW	H L L L	X X X L X X X H	BA_9 BA_8--BA_1 BA_0 PC+1	L L	H	If Test Register 0 is LOW: BA_0 - $BA_9 \rightarrow PC$ If Test Register 0 is HIGH: PC+1 $\rightarrow$ PC
	BTL_1	Branch on T_1 LOW	H L L H	X X L X X X H X	BA_9 BA_8--BA_1 BA_0 PC+1	L L	H	If Test Register 1 is LOW: BA_0 - $BA_9 \rightarrow PC$ If Test Register 1 is HIGH: PC+1 $\rightarrow$ PC
	BTL_2	Branch on T_2 LOW	H L H L	X L X X X H X X	BA_9 BA_8--BA_1 BA_0 PC+1	L L	H	If Test Register 2 is LOW: BA_0 - $BA_9 \rightarrow PC$ If Test Register 2 is HIGH: PC+1 $\rightarrow$ PC
	BTL_3	Branch on T_3 LOW	H L H H	L X X X H X X X	BA_9 BA_8--BA_1 BA_0 PC+1	L L	H	If Test Register 3 is LOW: BA_0 - $BA_9 \rightarrow PC$ If Test Register 3 is HIGH: PC+1 $\rightarrow$ PC
Miscellaneous Instructions	RTS	Return from Subroutine	L L L L	X X X X	Contents of the Stack Addressed by Read Pointer	L L	L	Pop the Stack
	FTCH	FETCH	L L H L	X X X X	PC+1	L L	L	PC+1 $\rightarrow$ PC

L = LOW Level
H = HIGH Level
X = Don't Care

Table 4-1. Instruction Sets for Macrologic Devices (Cont.)

EXERCISE 1 — UNCONDITIONAL BRANCHING IN 9408

The basic instruction of the 9408 Microprogram Sequencer is FETCH, where the Program Counter (PC) is incremented after each clock to select the address of the next microinstruction. This type of instruction is used for the machine to execute a sequence of consecutive microinstructions. To transfer control to execute a microinstruction other than the next one, a branch instruction has to be executed. *Figure 4-1* shows an example incorporating the BRV_0 unconditional branch instruction. Here locations 0 through 2 contain FETCH instructions and location 3 contains a BRV_0 to location 0. Execution of the BRV_0 instruction causes control word bits 16 to 23 (C16-C23) to be loaded into the 9408 program counter. In this case, the 9408 program counter is loaded with zero. The execution sequence of this microprogram will thus be 0, 1, 2, 3, 0, 1, 2, 3, 0, and so forth. *Figure 4-2* illustrates the BRV_1 instruction. The BRV_1 instruction is also an unconditional branch, but instead of loading the branch address from a field in the word, the 9408 selects the eight-input toggle switch setting as its next address. The execution sequence of this example will be o, x, where x = input switch settings.

EXERCISE 2 — CONDITIONAL BRANCHING IN 9408

Although the unconditional branch instructions add some flexibility to the sequencing of microinstructions, the machine still lacks any decision-making capability. This can be provided by the 9408 conditional branch instructions. The 9408 performs a test on one of four ALRS status flags (zero, negative, overflow and carry). If the test condition is met, the 9408 program counter will be loaded with the address represented by control word bits 16 to 23 (C16-C23). If the condition is not met, the program counter will be incremented by one (FETCH). *Figure 4-3* shows an example of the "Branch If Not Zero" instruction. In location 8, control bits 8 through 13 (C8-C13) generate the constant zero from the 9404 DPS. The ALRS source and destination registers are both selected to be register 0 and the ALRS instruction (C21-C23) is a load. The 9408 instruction field is FETCH. Thus execution of this microinstruction will cause binary zero to be loaded into register 0 of the ALRS. Location 9 control word bits 0 and 1 (C0-C1) selects the ALRS output register to source the 9404 input bus. The 9404 instruction field shows a logical left shift (C8-C13). Again ALRS register 0 is selected to be both the source and destination registers and the 9405A is going to execute an Inclusive OR (IOR) instruction. Carry control (C30) is a "1" which will cause a logic "0" to be the 9404 carry in during the left shift. This instruction causes the 9405A output register containing "0" as a result of instruction 8 to go through the DPS input bus, shift left one place, filling in the right most bit position with a logic "0". This data will then be inclusive ORed with the original contents of ALRS register 0, and the result will again be stored in ALRS register 0. The ALRS flags will be stored into the 9408 test register at the end of this instruction. Location A contains a "Branch If Not Zero" instruction. Here the 9408 examines if the ALRS ZERO flag is true or not. If it is not true, it will execute a branch to location A, which is a branch-to-itself loop. If the devices are operating correctly, the branch will not be taken and the next instruction in location B, BRV_0 to location 8, will be executed. The normal program execution sequence will therefore be 8, 9, A, B, 8, 9, etc.

EXERCISE 3 — ROTATE FUNCTION IN THE DEMONSTRATOR

As mentioned before, the shift and rotate functions in Macrologic devices are isolated from the ALRS and put into the DPS so that they can be shared with other Macrologic devices in the data path. In this example, a pattern is loaded from the input switches into a register in the DAR and rotated left continuously *(Figure 4-4)*. In location 0, control bits 2 to 6 select the DARs to load the DPS output bus into register 0 of its three registers. In addition, control bit 7 selects the M bus to display the contents of that register. The 9404 instruction field depicts a "Load Complement of K-Bus," which are the input switches. Control bit 29 disables the Carry In input for the 9407 so that the function $\overline{D} \rightarrow R_0$ is realized. The 9408 instruction is a FETCH. After execution of this instruction, the M-bus data LEDs should display a pattern the same as the input switch settings. In location 1, the 9407 output register contents are selected to source the DPS input bus by control bits 0 and 1 (C0 and C1). Control bits 8 through 13 depict a left shift function in the DPS. However, since rotate control bit C14 is set, the rotate left function is chosen. Control bits 2 through 7 and C29 again

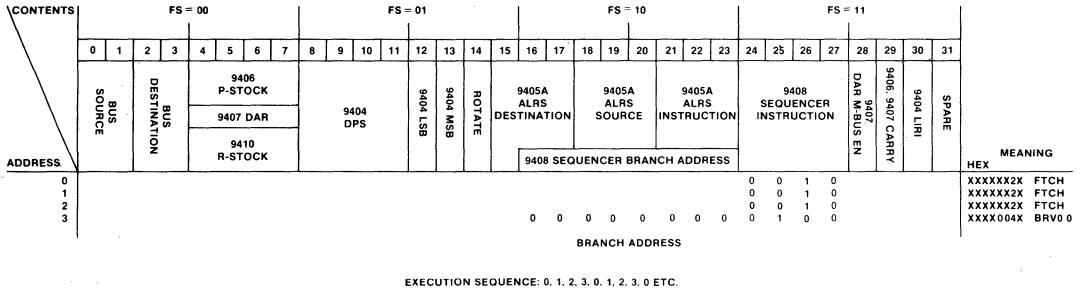


Fig. 4-1 Exercise 1a BRV0

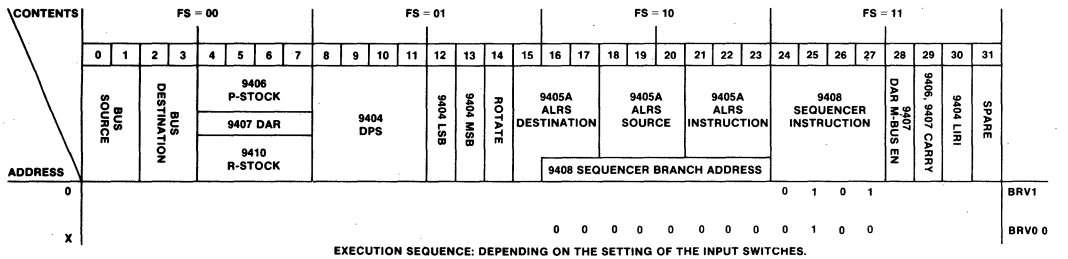


Fig. 4-2 Exercise 1b BRV1

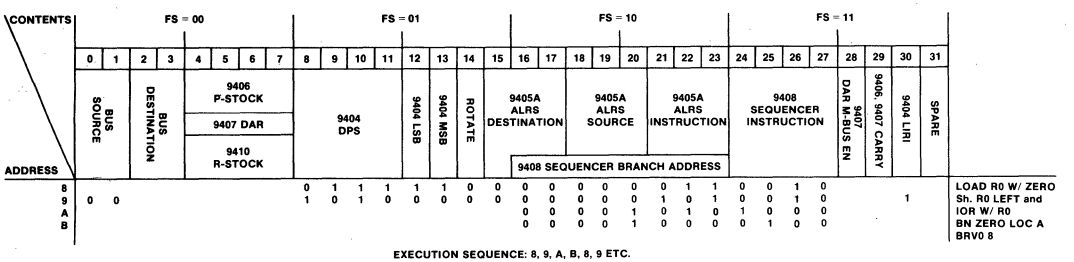


Fig. 4-3 Exercise 2 - Conditional Branch in 9408

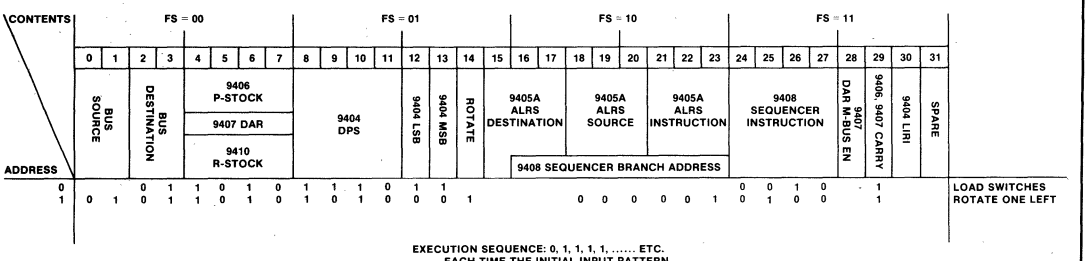


Fig. 4-4 Exercise 3 Rotate Function Using DAR and DPS

instruct the DAR to load the DPS output bus into register 0 of its three registers without any carry. The 9408 instruction is a BRV_0 to location 1 which causes this microinstruction to be executed again and again. Therefore this microprogram will hang in location 1 and, each time the Single Step switch is pressed, the M-bus LED will show the previous displayed pattern rotated one position to the left. The example illustrates the concept of iterative microinstruction execution. Often times in microprogramming, it is necessary to repeat executing a certain step until a test condition becomes true. In the Demonstrator architecture, the same microinstruction can contain instructions to the various Macrologic devices in the data path and simultaneously check for a test condition to become true.

EXERCISE 4 — TIMING LOOP EXAMPLE TO ILLUSTRATE BRANCH TO SUBROUTINE IN 9408

Just as in higher level language programming, the advantages of subroutining can be realized in microprogramming. The idea here, of course, is that the same block of microcode can be shared by several instruction sequences. This results in the overall reduction in the total number of microprogram memory words required by a design. Sometimes a subroutine may call another subroutine in its microinstruction sequence; this is called *nesting* of subroutines. The 9408 allows up to four levels of subroutine nesting. Exercise 4 illustrates a delay loop example with two levels of subroutine nesting *(Figure 4-5)*. It is essentially the same as Exercise 3. Instead of branching back to itself in location 1, it executes a branch to subroutine DELAY (location 37 on Appendix Microprogram examples). Upon return, location 2 contains a BRV_0 to location 1. The effect of executing this microprogram example is that the input pattern will still be rotated left one position at a time, but with a 1/2-second delay between each rotation so the program can be run and still produces a pattern noticeable to the human eye. Subroutine DELAY creates that 1/2-second delay. Subroutine DELAY calls subroutine TIMEY eight times; essentially, subroutine TIMEY generates a 60-millisecond delay. Microinstructions 41 to 44 constitute a timing loop where register 4 of the ALRS is loaded with zero initially, and is decremented by one each time until zero result is again reached. For each completion of this inner loop, a similar countdown is performed on register 3 in the outer loop. The calculation of the delay is shown as follows, assuming a clock rate of about 3.3 MHz.

Calculation of Delay Loop Timing

Subroutine TIMEY generates a delay of $2 + 256\ [2 + 3\ (256) + 1]$ clock cycles = 197378 CP. Assuming clock rate of 275 ns, delay = $0.3 \times 10\text{-}^6 \times 2 \times 10^5$ second = 0.06 second or 60 ms. Subroutine DELAY calls TIMEY and generates a delay of $1 + 8\ (3 + \text{TIMEY}) + 1$ CP $\cong$ 0.48 second or 500 ms.

EXERCISE 5 — TIMING WAVEFORM GENERATION

In microprogrammed system, it is sometimes necessary to generate a signal which goes HIGH once every 10th clock and stays LOW the rest of the time. Recall that in the microcontrol word definition, control bit 31 (C31) is used as a spare bit. In this exercise, C31 is used to generate the required timing signal. *Figure 4-6* shows a flow chart of the microprogram. The basic strategy is to use register 0 of the ALRS as a shift register. It is first loaded with the binary pattern 00001000; then the pattern is rotated left one place and the result stored back to register 0 of the ALRSs. A test is performed to see if the most significant bit of the result is set. If not, the pattern is rotated left one more place and checked again. Otherwise, control bit 31 (C31) is marked HIGH, register 0 of the ALRS is reloaded with the same starting binary pattern, and the procedure is repeated.

The actual microprogram is shown in *Figure 4-7*. The various fields of each control word are marked with the appropriate 1's and 0's. The blank entries are "don't care." In location 0, control bits 8 through 13 specify the instructions "Negative Sign Mask" on the least significant nibble and "Positive Sign Mask" on the most significant nibble of the 9404. The result is the generation of the binary constant 00001000 on the 9404 output bus. Control bits 15 through 23 specify the ALRS to load that result into register 0 of its eight registers. Control bits 24 through 27 instruct the 9408 to execute a FETCH so that Microinstruction 1 will be executed next. Control bit 31 is marked 0.

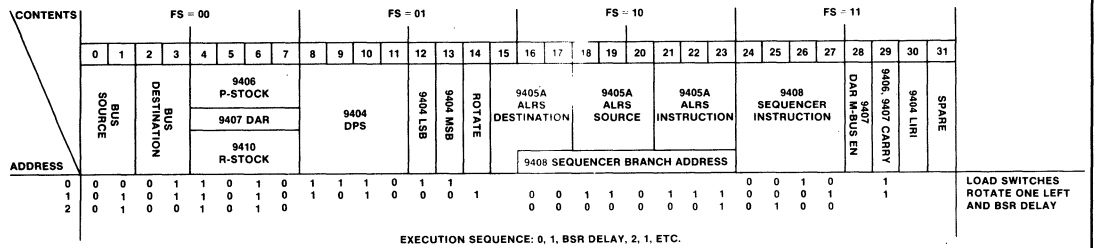


Fig. 4-5 Exercise 4 Timing Loop Example

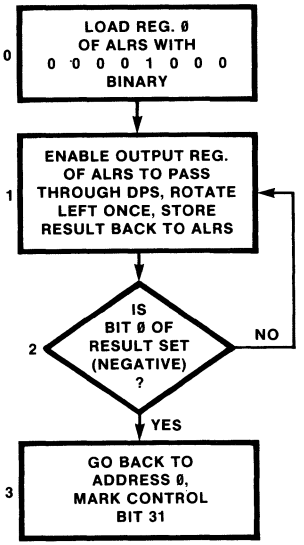


Fig. 4-6 Flowchart for Exercise 5

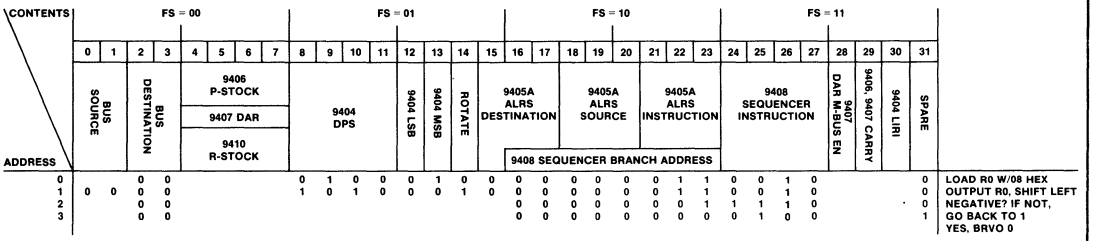


Fig. 4-7 Microprogram for Exercise 5

Execution of this microinstruction will cause the binary information 00001000 to be loaded into register 0 of the ALRS. The result is also strobed into the ALRS output register during the rising edge of the system clock.

In location 1, control bits 0 and 1 enable the ALRS output register to drive the D-bus of the DPS. Control bits 8 through 13 instruct the DPS to "shift left," control bit 14 specifies "rotate;" these combine to cause the information to be rotated left one place. Bits 16 through 23 command the ALRS to load the DPS outputs into register 0. Again control bits 24 through 27 specify a FETCH instruction for the 9408 and control bit 31 is left zero. Location 2 contains a "Branch if Negative Flag Not Set" 9408 conditional branch instruction specified by control bits 24 through 27. The branch address is location 1 specified by control bits 16 through 23. Recall that the ALRS status flags (carry, negative, overflow and zero) are strobed into the 9408 internal test registers at the end of the microcycle in which the ALRSs are activated. Thus, in this case, the negative status flag generated as a result of Microinstruction 2 is examined. If it is not set, the 9408 program counter will be loaded with the address represented by control word bits 16 to 23, i.e., location 1. If the test condition is not met, i.e., negative flag is set, the 9408 will execute a FETCH and go to location 3 and control bit 31 remains 0. Location 3 contains an unconditional branch instruction. Control word bits 24 to 27 specify a BRV_0 instruction to the 9408. The branch address (location 0) is specified by control word bits 16 to 23. Control bit 31 is marked 1. Going through the microprogram shows that location 2 and 3 will be executed four times before the negative flag is set, causing the microprogram to go to location 3. The execution sequence of the microprogram will be 0, 1, 2, 1, 2, 1, 2, 1, 2, 3, 0, 1, 2, 1, etc. Therefore, the microinstruction in location 3 is executed once every ten clock cycles and the desired waveform is generated.

Some very accurate timing can be generated in this manner, provided that the HIGH and LOW durations of the waveform are multiples of the basic clock period and, of course, that the clock period is precisely controlled. It is quite evident that the desired waveform could be achieved by a pure hardware approach (for example, a divide-by-10 counter). Therefore, what is the merit of using microprogramming? To generate more complicated timing may take more MSI, SSI packages than the corresponding microprogramming approach. Besides, the microprogramming approach is more highly structured and organized. Moreover, when one has to make a change, it is usually easier to modify the microprogram than to tear up gates and flip-flops.

EXERCISE 6 — SWITCHING CHARACTERISTICS MEASUREMENT USING ADDRESS SYNC

Examining the output waveform produced from Exercise 5 shows that there is a delay between the rising edge of the 10th clock pulse and when control bit 31 comes up HIGH. This is due to the propagation delay which is the sum of the clock to address output delay of the 9408 and the address access time (t_{AA}) of the microprogram memory shown in *Figure 4-8*. These values are given in the Macrologic Data Book to be typically 52 ns and 30 ns, respectively. The actual values of the propagation delays can be verified using the Address Sync feature of the Demonstrator. First of all, it is necessary to identify what happens on the oscilloscope. The Address Sync signal helps put a reference trace on the scope, and signals in the vicinity of the trace can be identified. In this case, Microinstruction 3 is a convenient point to synchronize to because it is only executed once every time around the program loop. To do that, put the program in Run mode and set the data input switches to 3. Each time Microinstruction 3 is executed, a HIGH pulse will be seen *(Figure 4-9)*.

The combination of the system clock (CP) and Address Sync traces helps deduce and identify the execution sequence in time on the scope. To measure the clock to address input propagation delay of the 9408, notice that the least significant 9408 address line (A_0) goes from LOW to HIGH during Microinstruction 3. Thus the interval between rising edge of the system clock and LOW to HIGH transition of A_0 can be measured as shown in *Figure 4-10*. Similarly, control bit 31 (C31) also comes up HIGH in Microinstruction 3. Therefore, the control memory address access time (t_{AA}) is

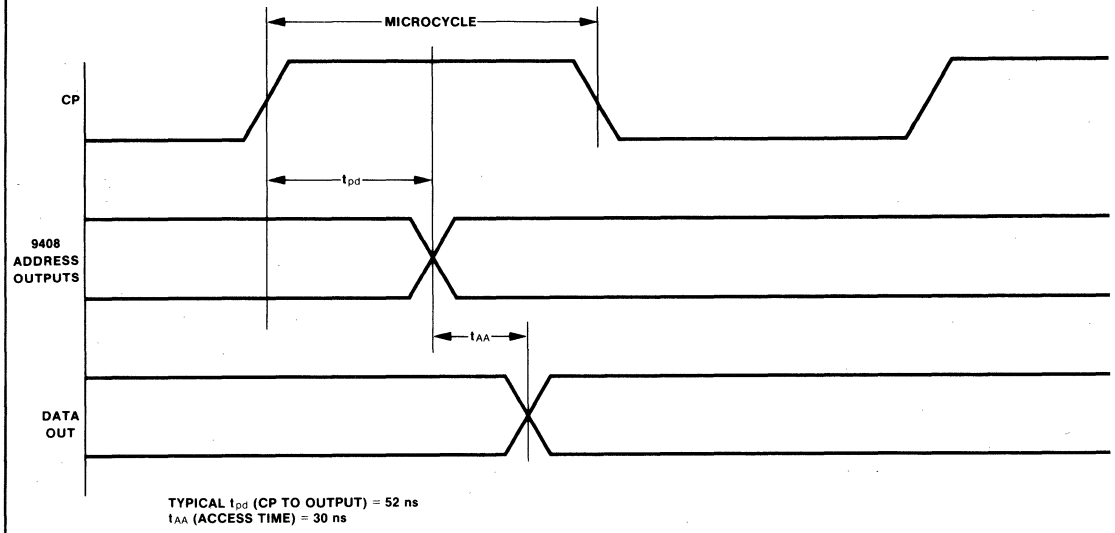


Fig. 4-8 Typical Propagation Delays

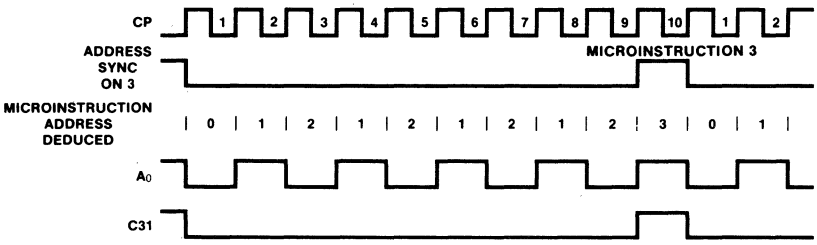


Fig. 4-9 Oscilloscope Waveforms

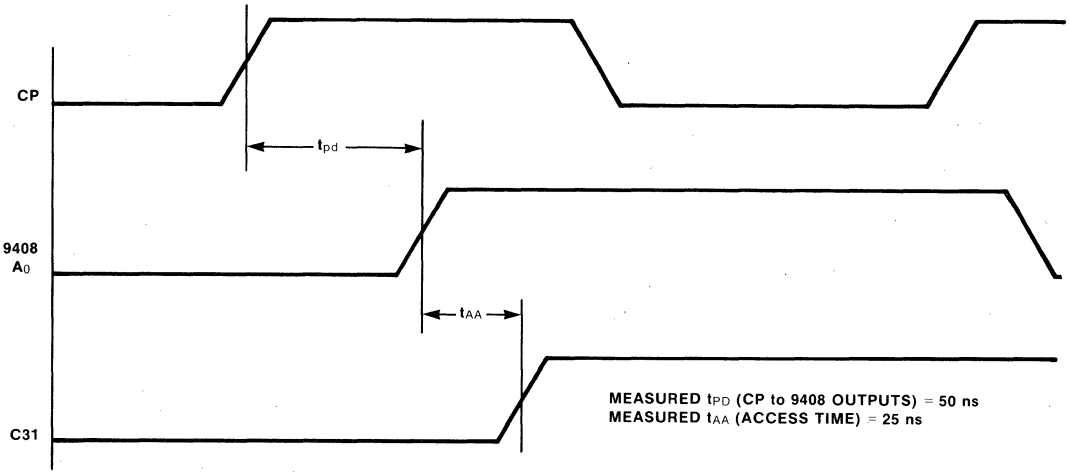


Fig. 4-10 Measurement of AC Parameters

measured as the interval between the rising edge of A_0 and the LOW to HIGH transition of C31. The measured values of the clock to output delay of the 9408 and the address access time of the microprogram memory are 50 and 25 ns, respectively.

Appendix A contains more microprogramming examples. The Multiway switches are used to select one of seven programs. These include:

1. A simple functional test of the various Macrologic devices.
2. An automatic pattern generation program which makes light blink.
3. A reaction time measurement program.
4. A multiplication program.

The user is encouraged to write his own microprogram examples. Blank programming sheets are provided in Appendix B. Appendix C is a sample programming format. Some suggestions are:

1. Measurement of some ac characteristics such as propagation delay of various Macrologic parts.
2. Ping pong games utilizing the LED display and input switches as paddles.
3. Using control bit C31 to create a 50-Hz square wave.
4. Use the P-stack to illustrate the Macro subroutine branching.
5. A division program, perhaps using the DARs registers.
6. A sorting program to order three or four words in descending numerical order in the ALRS register.

Exercises presented here are only representative and the user should write additional exercises to evaluate any specific parameter or feature in which he is interested.

APPENDICES

Appendix A
MICROASSEMBLY EXAMPLES

INSTRUCTIONS FOR RUNNING SAMPLE PROGRAM IN APPENDIX
LOAD in the desired programs.

Simple Function Test Program

1. Set Multiway switches to 001. Hit RESET and single step through it. Execution sequence should be 0, 1, 8, 9, A, B, C, D, E, F, 10, 11, 12, 8, 9, A, B and so forth. This tests the four ALU flags and the 9408 branch on one of these conditions.
2. This tests the 9406. Set address to 14 and single step through it. Monitor the M-bus as the 9406 carries through Branch, Fetch, Call and Return instructions.
3. The next test is for 9407. Set address to 1D and single step through it. Monitor the M-bus and observe the program loads in the input switches and left shift the patterns it alternates between 1E and 1F.
4. The next test is for 9410. Set address to 20 and single step through it. Monitor the various busses to observe the 9410 is loading and storing the data from input switches correctly.

Pattern Generation, Automatic
Set Multiway switches to "010." Set Data Select switches to "10" to look at M-bus. Run the program and observe LEDs in M-bus flash in some fixed patterns. This program calls the DELAY routine.

Reaction Time Routine
Set Multiway switches to "011." Set Data Select switches to "10." Hit Run. After a while, the least significant M-bus LED will come up. Flip any one of the input switches as soon as you can. The program will measure the delay between when the light comes up and when the user flips the switch, and display a binary number whose magnitude is proportional to that delay.

Multiplication Routine
Set Multiway switches to "111." Set Data Select switches to "00" to look at O-bus. Single step until address shows 5F. Input a 4-bit binary number through input switches. Hit Single Step. Address should show 60. Load in another number and hit Run. The O-bus LEDs will display the binary result.

```
**
**
**************************************************************************
**                                                                      **
**              MACROLOGIC DEMONSTRATION KIT MICROPROGRAM               **
**                                                                      **
**************************************************************************
**
**
** FIRST STEP : DEFINE THE MICROPROGRAM WORD FIELDS **
**
**
** DEVICE OUTPUT ENABLE SELECT FILED - 2 BITS **
**

* FIELD DEVOE  0,2
      ALUOE=00,  DAROE=01,  PSTKOE=10,  RSTKOE=11
**
** DEVICE EXECUTE SELECT FIELD - 2 BITS **
**

* FIELD  DEVEX  2,2
      EXTEX=00,  DAREX=01,  PSTKEX=10,  RSTKEX=11
**
** 9406 INSTRUCTION FIELD - 2 BITS (OVERLAPPING) **
**

* FIELD PSTKINST  6,2
      RETURN=00,  BRANCH=01,  CALL=10,  FETCH=11
**
** 9407 INSTRUCTION FIELD 1 - 3 BITS (OVERLAPPING) **
**

* FIELD DARINST1   4,3
      R0+D->R0=000,  R0+D->R1=001,  R0+D->R2=010,
      R1+D->R1=011,  R2+D->R2=110,  D->R0=101,
      D->R1=111,     D->R2=100     ,  DEFAULT=101
**
** 9407 INSTRUCTION FIELD 2, X-BUS SOURCE SELECT - 1 BIT **
**

* FIELD DARINST2  7,1
      REG->X-BUS=0,  ADDER->X-BUS=1
**
** 9410 ADDRESS FIELD - 4 BITS (OVERLAPPING) **
**

* FIELD RSTKADDR  4,4
      A00=0000,  A01=0001,  A02=0010,  A03=0011,
      A04=0100,  A05=0101,  A06=0110,  A07=0111,
      A08=1000,  A09=1001,  A10=1010,  A11=1011,
      A12=1100,  A13=1101,  A14=1110,  A15=1111
**
**
** 9404 INSTRUCTION FIELD - 4 MSB **
**

* FIELD  DPSINST 8,4
   BYTEMASK=0000,  MINUS-TWO=0001,  MINUS-ONE=0001, DMASK0=0010,
```

```
   DMASK1=0011,   SIGNMASK=0100,  KMASK1=0101,   PASS=0110,
   ONE=0111,     ZERO=0111,     KSIGN=1000,    DSIGN=1001,
   DSHL=1010,     KSHL=1010,     DSHR=1011,     KSHR=1100,
   KMASKO=1101,   COMPLEMENT=1110, DEFAULT=0110
**
** LSB OF 9404 INSTRUCTION FOR LEAST SIGNIFICANT BYTE **
**

* FIELD  DPSLSB  12,1
     LS1=1,   MINUS-ONE=1,   KSHL=1,   ZERO=1 ,  KIN=1
**
** MSB OF 9404 INSTRUCTION FOR MOST SIGNIFICANT BYTE **
**

* FIELD  DPSMSB  13,1
     MINUS-TWO=1,  ONE=1,  KSHL=1   ,  MINUS-ONE=1,
     MS1=1,   ZERO=1 , KIN=1
**
** SHIFT W/ EXTERNAL CARRY OR ROTATE SECLECT FIELD - 1 BIT **
**

* FIELD ROTATE  14,1
      SHIFT=0,  ROTATE=1,  DEFAULT=0
**
** NEXT ADDRESS FIELD - 8 BITS ( OVERLAPPING W/9405A)  **
**

* FIELD ADDRESS  15,9
**
** 9405A DESTINATION REG FIELD - 3 BITS **
**

* FIELD DEST-REG  15,3
      D0=000,  D1=001,  D2=010,  D3=011,
      D4=100,  D5=101,  D6=110,  D7=111
**
** 9405A SOURCE REGISTER FIELD - 3 BITS  **
**

* FIELD SOURCE-REG  18,3
      S0=000,  S1=001,  S2=010,  S3=011,
      S4=100,  S5=101,  S6=110,  S7=111
**
** 9405A INSTRUCTION FIELD - 3 BITS  **
**

* FIELD ALRSINST  21,3
     R+D+1  =000,  ADD   =001,  AND=010,  LOAD=011
     OUTPUT =100,  IOR   =101,  XOR=110,  COMP=111
     INCR=000,   DECR=001, DEFAULT=100
**
** 9408 INSTRUCTION FIELD - 4 BITS **
**

* FIELD  SEQOP  24,4
     RTS=0000,  BSR=0001,  FTCH=0010, BMW=0011
     BRV0=0100, BRV1=0101, BRV2=0110, BRV3=0111
     BTL0=1000, BTL1=1001, BTL2=1010, BTL3=1011
     BTH0=1100, BTH1=1101, BTH2=1110, BTH3=1111,
     BCARRY=1011,  BZERO=1100,   BNEG=1010,  BNNEG=1110,
     BOVFL=1001, BNOVFL=1101, BNZERO=1000, BNCARY=1111,  DEFAULT=0010
```

```
**
**   DAR M-BUS OUTPUT INHIBIT
**


* FIELD DARXE 28,1
       INH-DARXE=1
**
** CARRY IN CONTROL FOR 9406,9407


* FIELD CARYIN 29,1
       FETCH=0, DEFAULT=1
**
** CARRY IN CONTROL FOR SHIFTS
**


* FIELD LIRI 30,1
       SH1=0, DEFAULT=1
**
** SPARE FIELD
**


* FIELD SPARE 31,1

       MARK=1

**
** NEXT STEP : WRITE YOUR MICROPROGRAM USING THE MNEMONICS            **
**             YOU  HAVE PREVIOUSLY DEFINED. EACH LINE OF CODE        **
**             REPRESENTS ONE MICRO-INSTRUCTION.                      **
**
*************************************************************************
**

* PROGRAM    HEX,HEX

**
*************************************************************************
**
** MACROLOGIC LEARNING KIT RESIDENT DEMONSTRATION  PROGRAMS
**
*************************************************************************
BEGIN      DAREX,D->RO,BYTEMASK,BEGIN,BMW          PROGRAM SELECT
    0 1A000036

           TEST,BRVO                              SYSTEM TEST PROGRAM
    1 0A600846

           PAT0,BRVO                               PATTERN GEN. AUTO
    2 0A602346

           REACT,BRVO                              REACTION TIME
    3 0A604546

           PONG,BRVO                               PING PONG
    4 0A605A46

           SHOOT,BRVO                              SHOOTING
```

```
     5 0A6058A6

          EMUL,BRVO                                   EMULATE PROGRAM
     6 0A605C46

          MULTI,BRVO                                 MULTIPLICATION
     7 0A605D46

*****************************************************************
**     SIMPLE FUNCTIONAL TEST  PROGRAM                         **
*****************************************************************
TEST      ZERO,DO,LOAD                                TEST 9408+9404+9405
     8 0A7C0326

          DSHL,ROTATE,DO,SO,IOR
     9 0AA20526

ERR1      BNZERO,ERR1                                  ZERO FLAG
     A 0A600A86

          COMPLEMENT,D1,LOAD
     B 0AE04326

          ONE,D2,S1,ADD
     C 0A748926

ERR2      BNCARY,ERR2                                 CARRY FLAG
     D 0A600DF6

          SIGNMASK,LS1,D2,S2,XOR
     E 0A489626

ERR3      BNNEG,ERR3                                   NEGATIVE FLAG
     F 0A600FE6

          COMPLEMENT,D3,LOAD
    10 0AE0C326

          ONE,D3,S3,ADD
    11 0A74D926

          TEST,BOVFL                                   OVERFLOW FLAG
    12 0A600896

ERR4      BRVO,ERR4
    13 0A601346

PSTKTST   PSTKEX,BRANCH,ONE                                 9406 TEST
    14 29740426

          PSTKOE,PSTKEX,FETCH
    15 AB600422

          PSTKOE,PSTKEX,FETCH
    16 AB600422

          PSTKOE,PSTKEX,CALL,KMASK1
    17 AA500426

          PSTKOE,PSTKEX,FETCH
    18 AB600422
```

```
                PSTKOE,PSTKEX,CALL,KMASK1
    19 AA500426

                PSTKOE,PSTKEX,FETCH
    1A AB600422

                PSTKOE,PSTKEX,RETURN
    1B A8600426

                PSTKOE,PSTKEX,RETURN,BRVO,PSTKTST
    1C A8601446

                DAREX,D->R1,COMPLEMENT,KIN                    9407 TEST
    1D 1EEC0426

DARTST      DAROE,DAREX,R1+D->R1                              SHIFT LEFT
    1E 56600426

                DAROE,EXTEX, D->R1,COMPLEMENT,BRVO,DARTST
    1F 4EE01E46

                RSTKEX,A10,COMPLEMENT,LS1,MS1,D5,LOAD
    20 3AED4326

RSTKTST     RSTKOE,COMPLEMENT,D5,S5,COMP
    21 CAE16F26

                RSTKEX,A05,COMPLEMENT,BRVO,RSTKTST
    22 35E02146

*************************************************************
** (2). PATTERN GENERATION , AUTOMATIC                     **
*************************************************************
PAT0        DAREX,D->R0,SIGNMASK,D1,LOAD                      PAT. 1
    23 1A404326

                DAROE,DAREX,D->R0,COMPLEMENT,DELAY,BSR
    24 5AE03716

                DAROE,DAREX,D->R0,DSHR,DELAY,BSR              PATTERN 2
    25 5AB03716

                DAROE,DAREX,D->R0,DSHR,DELAY,BSR
    26 5AB03716

                DAREX,R0+D->R0,PASS,DELAY,BSR                 PAT. 3
    27 10603716

                DAROE,DAREX,D->R0,DSHL,ROTATE,DELAY,BSR       PAT. 4
    28 5AA23716

                DAREX,D->R0,SIGNMASK,DELAY,BSR                PAT 5
    29 1A403716

                DAROE,DAREX,D->R0,DSHR,LS1,MS1,DELAY,BSR      PAT 6
    2A 5ABC3716

                DAROE,DAREX,D->R0,COMPLEMENT,DELAY,BSR        PAT 7
    2B 5AE03716

                DAROE,DAREX,D->R0,DSHL,ROTATE,DELAY,BSR       PAT 8
```

```
   2C 5AA23716

          DAROE,DAREX,D->RO,COMPLEMENT,DELAY,BSR          PAT 9
   2D 5AE03716

          DAREX,D->RO,ZERO,DELAY,BSR                      PAT 10
   2E 1A7C3716

          DAROE,DAREX,D->RO,COMPLEMENT,DELAY,BSR          PAT 11
   2F 5AE03716

          DAREX,D->RO,BYTEMASK,MS1,DELAY,BSR              PAT 12
   30 1A043716

          DAROE,DSHR,ROTATE,D1,LOAD
   31 4AB24326

          DAREX,D->RO,DSHR,ROTATE,DELAY,BSR               PAT 13
   32 1AB23716

          DAROE,DAREX,D->RO,COMPLEMENT,DELAY,BSR          PAT 14
   33 5AE03716

          DAROE,DSHR,ROTATE,D1,LOAD
   34 4AB24326

          DAREX,D->RO,DSHR,ROTATE,DELAY,BSR               PAT 15
   35 1AB23716

          PAT0,BRV0                                       LOOP
   36 0A602346

**  DELAY ROUTINE
DELAY     SIGNMASK,MS1,D2,LOAD                 LOAD 8
   37 0A448326

DELOP     MINUS-ONE,D2,S2,ADD                  DECREMENT 1
   38 0A1C9126

          DOUT,BZERO                           UNTIL ZER
   39 0A603CC6

          TIMEY,BSR                            CALL TIME OUT
   3A 0A603D16

          DELOP,BRV0                           REPEAT
   3B 0A603846

DOUT      RTS                                  RETURN
   3C 0A600406

** TIME OUT ROUTINE
TIMEY     SIGNMASK,LS1,D3,LOAD                 LOAD CT
   3D 0A48C326

T1        MINUS-ONE,D3,S3,ADD                  INC BY 1
   3E 0A1CD926

          T3,BZERO                             UNTIL ZERO
   3F 0A6044C6
```

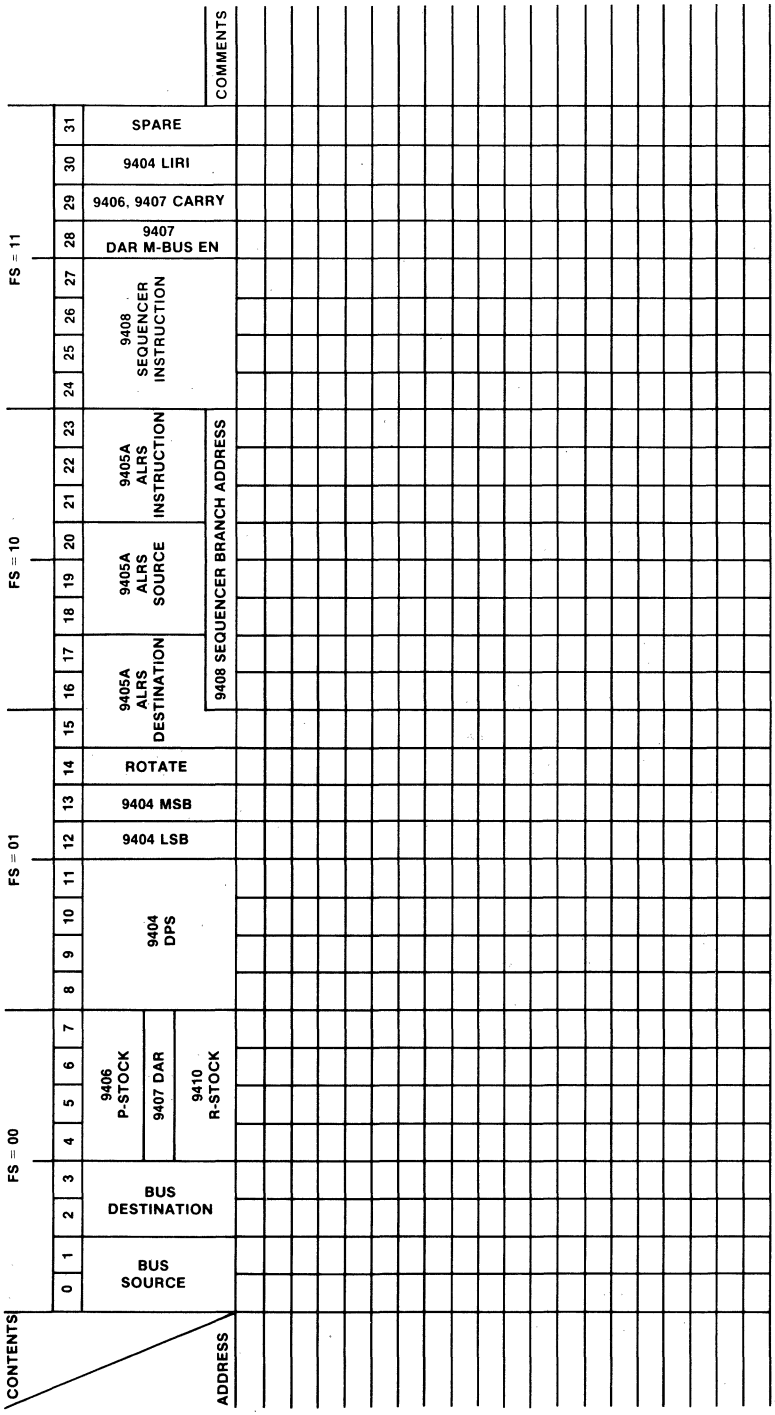
```
          SIGNMASK,LS1,D4,LOAD                    INNER LOOP
   40 0A490326

T2        MINUS-ONE,D4,S4,ADD                     DEC BY 1
   41 0A1D2126

          T1,BZERO
   42 0A603EC6

          T2,BRVO                                 REPEAT
   43 0A604146

T3        RTS                                     RETURN TO CALL PT
   44 0A600406

*************************************************************
** (3). GAME TO MEASURE REACTION TIME OF HOMO-SAPIEN  **
*************************************************************
REACT     DAREX,D->RO,BYTEMASK,MULRAM,BSR         GENERATE RANDOM NO
   45 1A005216

          ZERO,D7,LOAD                            CLR REG 7
   46 0A7DC326

COUNT     TIMEY,BSR                               DELAY
   47 0A603D16

          ONE,D2,S2,ADD                           INC COUNT
   48 0A749126

          COUNT,BNCARY                            TIME OUT ?
   49 0A6047F6

          KMASK1,D6,LOAD                          YES, LOAD SW
   4A 0A518326

          DAREX,D->RO,MINUS-TWO                   OUTPUT 1 LIGHT
   4B 1A140426

AGAIN     TIMEY,BSR                               INTRODUCE DELAY
   4C 0A603D16

          ONE,D7,S7,ADD                           INC COUNT
   4D 0A75F926

          KMASK1,D5,S6,XOR                        TEST IF SW CHANGED
   4E 0A517626

          AGAIN,BZERO                             REPEAT IF NOT
   4F 0A604CC6

          D7,S7,OUTPUT                            OUTPUT COUNT
   50 0A61FC26

FINISH    DAREX,D->RO,COMPLEMENT,FINISH,BRVO      DISPLAY REACTION T
   51 1AE05146

**SUBROUTINE MULRAM : GENERATES 8 BIT RANDOM NUMBER USING THE
**RESIDUE METHOD;IN THE FORM SEED * U(N) = U(N+1) WHERE
**                          REG0   REG1   REG2 IN ALRS
MULRAM    SIGNMASK,MS1,D3,LOAD                     LOAD I WITH 8
```

```
   52 0A44C326

          DSHR,D0,LOAD                              GENERATE 4
   53 0AB00326

          D3,S3,OUTPUT                              PUT 8 IN D-REG
   54 0A60DC26

          DSHL,D0,S0,ADD                            GENERATE 20
   55 0AA00126

          MINUS-ONE,D0,S0,ADD                       GENERATE SEED=19
   56 0A1C0126

          D1,S2,OUTPUT                               TRANSFER NEW U(N)
   57 0A605426

          BSR,MULT2                                   MULTIPLY
   58 0A606316

           RTS                                         RETURN
   59 0A600406

PONG      PONG,BRV0
   5A 0A605A46

$HOOT     SHOOT,BRV0
   5B 0A605B46

EMUL      EMUL,BRV0
   5C 0A605C46

MULTI     SIGNMASK,MS1,D3,LOAD                      LOAD 8
   5D 0A44C326

          ZERO,D2,LOAD                              CLR RESULT REG
   5E 0A7C8326

          DAREX,D->R0,KMASK0,MS1,D0,LOAD            LOAD NIBBLE
   5F 1AD40326

          PSTKEX,BRANCH,KMASK0,MS1,D1,LOAD          FROM SWITCHES
   60 29D44326

          MULT2,BSR                                   MULTIPLY
   61 0A606316

MULFIN    BRV0,MULFIN
   62 0A606246

***  SUBROUTINE MULT2 DOES 8BIT MULTIPLICATION
***   CALLING SEQUENCE: LOAD REG 3 W/8, CLR REG 2,
***   PUT MULTIPLIER AND MULTIPLICAND IN REG 0 AND 1 RESPECTIVELY,
***   THE PRODUCT WILL BE IN REG 2
MULT2     MULT1,BNNEG
   63 0A6065E6

          DAR0E,PASS,D2,S2,ADD
   64 4A609126

MULT1     MINUS-ONE,D3,S3,ADD                       DECREMENT COUNT
```

```
   65 0A1CD926

          MULTEXIT,BZERO                          FINISHED ?
   66 0A606BC6

          D2,S2,OUTPUT                            SET UP PARTIAL RESULT
   67 0A609426

          DSHL,D2,LOAD                            CONTINUE
   68 0AA08326

          PSTKOE,PSTKEX,BRANCH,DSHL,D1,LOAD
   69 A9A04326

          BRVO,MULT2
   6A 0A606346

MULTEXIT  D2,S2,OUTPUT,RTS                        DISPLAY RESULT AND RETURN
   6B 0A609406

*         END


*****   0 ERRORS *****
```

```
 0 1A000036
 1 0A600846
 2 0A602346
 3 0A604546
 4 0A605A46
 5 0A605B46
 6 0A605C46
 7 0A605D46
 8 0A7C0326
 9 0AA20526
 A 0A600A86
 B 0AE04326
 C 0A748926
 D 0A600DF6
 E 0A489626
 F 0A600FE6
10 0AE0C326
11 0A74D926
12 0A600896
13 0A601346
14 29740426
15 AB600422
16 AB600422
17 AA500426
18 AB600422
19 AA500426
1A AB600422
1B A8600426
1C A8601446
1D 1EEC0426
1E 56600426
1F 4EE01E46
20 3AED4326
21 CAE16F26
22 35E02146
23 1A404326
24 5AE03716
25 5AB03716
26 5AB03716
27 10603716
28 5AA23716
29 1A403716
2A 5ABC3716
2B 5AE03716
2C 5AA23716
2D 5AE03716
2E 1A7C3716
2F 5AE03716
30 1A043716
31 4AB24326
32 1AB23716
33 5AE03716
34 4AB24326
35 1AB23716
36 0A602346
37 0A448326
38 0A1C9126
39 0A603CC6
3A 0A603D16
3B 0A603846
3C 0A600406
3D 0A48C326
3E 0A1CD926
3F 0A6044C6
40 0A490326
41 0A1D2126
42 0A603EC6
43 0A604146
44 0A600406
45 1A005216
46 0A7DC326
47 0A603D16
48 0A749126
49 0A6047F6
4A 0A518326
4B 1A140426
4C 0A603D16
4D 0A75F926
4E 0A517626
4F 0A604CC6
50 0A61FC26
51 1AE05146
52 0A44C326
53 0AB00326
54 0A60DC26
55 0AA00126
56 0A1C0126
57 0A605426
58 0A606316
59 0A600406
5A 0A605A46
5B 0A605B46
5C 0A605C46
5D 0A44C326
5E 0A7C8326
5F 1AD40326
60 29D44326
61 0A606316
62 0A606246
63 0A6065E6
64 4A609126
65 0A1CD926
66 0A606BC6
67 0A609426
68 0AA08326
69 A9A04326
6A 0A606346
6B 0A609406
```

Appendix B
SAMPLE PROGRAMMING SHEETS

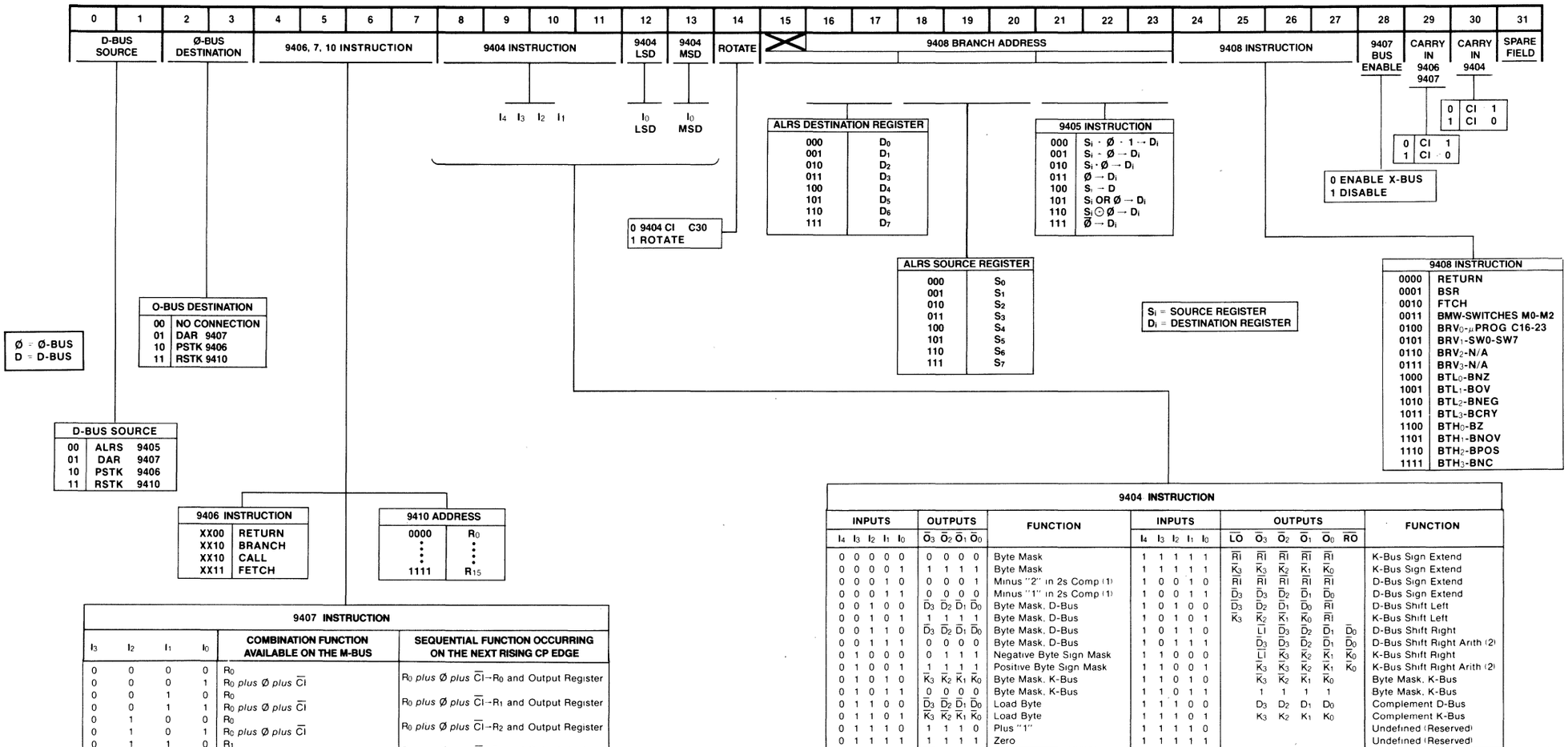


CONTENTS / ADDRESS	FS = 00								FS = 01								FS = 10								FS = 11								COMMENTS
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
	BUS SOURCE		BUS DESTINATION		9406 P-STOCK / 9407 DAR / 9410 R-STOCK				9404 DPS				9404 LSB	9404 MSB	ROTATE		9405A ALRS DESTINATION			9405A ALRS SOURCE		9405A ALRS INSTRUCTION			9408 SEQUENCER INSTRUCTION				9407 DAR M-BUS EN	9406, 9407 CARRY	9404 LIRI	SPARE	
																	9408 SEQUENCER BRANCH ADDRESS															

CONTENTS / ADDRESS	FS = 00								FS = 01								FS = 10								FS = 11								
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
	BUS SOURCE		BUS DESTINATION		9406 P-STOCK 9407 DAR 9410 R-STOCK				9404 DPS				9404 LSB	9404 MSB	ROTATE	9405A ALRS DESTINATION			9405A ALRS SOURCE			9405A ALRS INSTRUCTION			9408 SEQUENCER INSTRUCTION				9407 DAR M-BUS EN	9406, 9407 CARRY	9404 LIRI	SPARE	
ADDRESS																	9408 SEQUENCER BRANCH ADDRESS																COMMENTS

CONTENTS / ADDRESS	FS = 00								FS = 01								FS = 10								FS = 11								COMMENTS
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
	BUS SOURCE		BUS DESTINATION		9406 P-STOCK / 9407 DAR / 9410 R-STOCK				9404 DPS				9404 LSB	9404 MSB	ROTATE	9405A ALRS DESTINATION	9405A ALRS SOURCE				9405A ALRS INSTRUCTION			9408 SEQUENCER INSTRUCTION				9407 DAR M-BUS EN	9406, 9407 CARRY	9404 LIRI	SPARE	
																	9408 SEQUENCER BRANCH ADDRESS															

APPENDIX C
MACROLOGIC DEMONSTRATOR MICROWORD FORMAT



9404 INSTRUCTION

INPUTS I4	I3	I2	I1	I0	OUTPUTS $\overline{O_3}$	$\overline{O_2}$	$\overline{O_1}$	$\overline{O_0}$	FUNCTION
0	0	0	0	0	0	0	0	0	Byte Mask
0	0	0	0	1	1	1	1	1	Byte Mask
0	0	0	1	0	0	0	0	1	Minus "2" in 2s Comp (1)
0	0	0	1	1	0	0	0	0	Minus "1" in 2s Comp (1)
0	0	1	0	0	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$	Byte Mask, D-Bus
0	0	1	0	1	1	1	1	1	Byte Mask, D-Bus
0	0	1	1	0	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$	Byte Mask, D-Bus
0	0	1	1	1	0	0	0	0	Byte Mask, D-Bus
0	1	0	0	0	0	1	1	1	Negative Byte Sign Mask
0	1	0	0	1	1	1	1	1	Positive Byte Sign Mask
0	1	0	1	0	$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$	Byte Mask, K-Bus
0	1	0	1	1	0	0	0	0	Byte Mask, K-Bus
0	1	1	0	0	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$	Load Byte
0	1	1	0	1	$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$	Load Byte
0	1	1	1	0	1	1	1	0	Plus "1"
0	1	1	1	1	1	1	1	1	Zero

INPUTS I4	I3	I2	I1	I0	OUTPUTS $\overline{LO}$	$\overline{O_3}$	$\overline{O_2}$	$\overline{O_1}$	$\overline{O_0}$	$\overline{RO}$	FUNCTION
1	1	1	1	1	$\overline{R_1}$	$\overline{R_1}$	$\overline{R_1}$	$\overline{R_1}$	$\overline{R_1}$		K-Bus Sign Extend
1	1	1	1	1	$\overline{K_3}$	$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$		K-Bus Sign Extend
1	0	0	1	0	$\overline{R_1}$	$\overline{R_1}$	$\overline{R_1}$	$\overline{R_1}$	$\overline{R_1}$		D-Bus Sign Extend
1	0	0	1	1	$\overline{D_3}$	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$		D-Bus Sign Extend
1	0	1	0	0	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$	$\overline{R_1}$		D-Bus Shift Left
1	0	1	0	1	$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$	$\overline{R_1}$		K-Bus Shift Left
1	0	1	1	0		$\overline{L_1}$	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$	D-Bus Shift Right
1	0	1	1	1		$\overline{D_3}$	$\overline{D_3}$	$\overline{D_2}$	$\overline{D_1}$	$\overline{D_0}$	D-Bus Shift Right Arith (2)
1	1	0	0	0		$\overline{L_1}$	$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$	K-Bus Shift Right
1	1	0	0	1		$\overline{K_3}$	$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$	K-Bus Shift Right Arith (2)
1	1	0	1	0		$\overline{K_3}$	$\overline{K_2}$	$\overline{K_1}$	$\overline{K_0}$		Byte Mask, K-Bus
1	1	0	1	1		1	1	1	1		Byte Mask, K-Bus
1	1	1	0	0		D_3	D_2	D_1	D_0		Complement D-Bus
1	1	1	0	1		K_3	K_2	K_1	K_0		Complement K-Bus
1	1	1	1	0							Undefined (Reserved)
1	1	1	1	1							Undefined (Reserved)

9407 INSTRUCTION

I_3	I_2	I_1	I_0	COMBINATION FUNCTION AVAILABLE ON THE M-BUS	SEQUENTIAL FUNCTION OCCURRING ON THE NEXT RISING CP EDGE
0	0	0	0	R_0	
0	0	0	1	R_0 *plus* Ø *plus* $\overline{CI}$	R_0 *plus* Ø *plus* $\overline{CI}$→R_0 and Output Register
0	0	1	0	R_0	
0	0	1	1	R_0 *plus* Ø *plus* $\overline{CI}$	R_0 *plus* Ø *plus* $\overline{CI}$→R_1 and Output Register
0	1	0	0	R_0	
0	1	0	1	R_0 *plus* Ø *plus* $\overline{CI}$	R_0 *plus* Ø *plus* $\overline{CI}$→R_2 and Output Register
0	1	1	0	R_1	
0	1	1	1	R_1 *plus* Ø *plus* $\overline{CI}$	R_1 *plus* Ø *plus* $\overline{CI}$→R_1 and Output Register
1	0	0	0	R_2	
1	0	0	1	Ø *plus* $\overline{CI}$	Ø *plus* $\overline{CI}$→R_2 and Output Register
1	0	1	0	R_0	
1	0	1	1	Ø *plus* $\overline{CI}$	Ø *plus* $\overline{CI}$→R_0 and Output Register
1	1	0	0	R_2	
1	1	0	1	R_2 *plus* Ø *plus* $\overline{CI}$	R_2 *plus* Ø *plus* $\overline{CI}$→R_2 and Output Register
1	1	1	0	R_1	
1	1	1	1	Ø *plus* $\overline{CI}$	Ø *plus* $\overline{CI}$ →R_1 and Output Register

FAIRCHILD

Printed in U.S.A./331-70-0003-038 5C January 1979